Escuela Politécnica Superior

Centralita virtual sobre VoIP

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Adrián José Gosálvez Maciá

Tutor/es:
Domingo Gallardo López

Universitat d'Alacant
Universidad de Alicante

Septiembre 2018

Contexto

La idea del presente proyecto surgió cuando empecé a descubrir el mundo de la voz sobre IP (VoIP) y, después de investigar por mi cuenta, descubrí que para realizar una instalación de una centralita virtual telefónica auto-administrada en una empresa se necesitaban conocimientos avanzados de esta rama de las telecomunicaciones.

La telefonía sigue los pasos de la televisión, (hace unos años que nos despedimos de la analógica) y pronto pasará a ser digital también, de hecho, la compañía Movistar al ponernos los routers de fibra óptica nos está proporcionando la telefonía VoIP, aunque vuelven a realizar la conversión para los teléfonos analógicos que tenemos en casa de toda la vida. Una vez se realice el cambio, estarán preparados y ya tendrán todo montado.

Con esto quiero decir que el cambio es inminente y es una de las razones por las cuales me ha impulsado a realizar un proyecto sobre esta área de las telecomunicaciones.

Agradecimientos

En primer lugar, quiero agradecer a mi tutor por aceptar la propuesta del proyecto y por guiarme todos estos meses durante la realización del mismo, ha sido un placer.

También me gustaría agradecer a mi familia por estar aguantando esas charlas y explicaciones que obviamente, les sonaba a "chino" cada vez que empezaba a hablar de software y VoIP, sin embargo, me dedicaban su tiempo, que es lo que más valoro.

A mi pareja por el apoyo, los ánimos y por su granito de arena que ha ido haciendo fuerza día a día.

La innovación distingue a los líderes de los seguidores.

Steve Jobs

Índice

CONTEXTO **2**

AGRADECIMIENTOS **3**

ÍNDICE **5**

ÍNDICE DE ILUSTRACIONES **8**

1. INTRODUCCIÓN **11**

2. ESTADO DEL ARTE **12**

2.1 FREEPBX 12

2.2 ELASTIX 13

2.3 TRIXBOX 13

2.4 ASTERISKNOW 14

2.5 CONCLUSIONES 14

3. OBJETIVOS **16**

3.1 CENTRALITA VIRTUAL (PBX) 16

3.2 ADMINISTRACIÓN 16

4. TECNOLOGÍAS **17**

4.1 ASTERISK 17

4.1.1 PROTOCOLO SIP 17

4.1.2 PROTOCOLO RTP 18

4.1.3 DIALPLAN 18

4.1.3.1 ¿Qué significa dialplan configurado en Real Time? *19*

4.1.3.2 ¿Cómo conecto un teléfono para llamar mediante Asterisk? *20*

4.1.3.3 ¿Qué es un Trunk SIP? *20*

4.1.3.4 ¿Cómo funciona el flujo de llamadas en Asterisk? *21*

4.2 RUBY ON RAILS 21

4.3 PHP 22

4.4 MYSQL 22

5. METODOLOGÍA **23**

5.1 SCRUM Y KANBAN 23

5.2 TABLERO: TRELLO 23

5.2.1. Columnas del tablero *25*

5.2.2. Tarjetas *26*

5.3 FLUJO DE TRABAJO: GITFLOW 27

5.4 CONTROL DE VERSIONES E INTEGRACIÓN CONTINUA 28

5.4.1 GitHub *28*

5.4.2 Travis CI *28*

6. FUNCIONALIDADES **31**

6.1 ROL ADMINISTRADOR DEL SISTEMA 31

6.1.1 Gestión de usuarios *31*
6.1.2 Crear extensiones *31*
6.1.3 Crear departamentos *32*
6.1.4 Numeración *33*
6.1.4.1 Interna *33*
6.1.4.2 Entrante *34*
6.1.4.3 Saliente *35*
6.1.5 Locuciones *35*
6.1.6 Horarios *36*
6.1.7 Menú de llamada (IVR) *37*
6.1.8 Estadísticas *39*
6.2 Rol Trabajador 41
6.2.1 Control de fichado *41*
6.2.2 Desvío de extensión *43*
6.2.3 Incidencias (Mini CRM) *43*

7. ARQUITECTURA 45
7.1 Capa de presentación 45
7.2 Capa Intermedia 46
7.3 Capa Asterisk 48
7.4 Ejempos 48
7.4.1 Creación de extensiones *48*
7.4.2 Creación de horario *49*
7.4.3 Creación de menú selectivo *53*
7.5 Sincronización entre capas 56

8. SEGURIDAD 58
8.1 ¿Por qué atacan Asterisk? 58
8.2 Servidor 59
8.3 Asterisk 59

9. TESTING 61
9.1 Pruebas unitarias con Ruby on Rails 61
9.2 Pruebas de funcionales: Raspberry PI 62

10. INSTALACIÓN 63
10.1 Introducción 63
10.2 Instalación del servidor 63
10.3 Instalación de Asterisk PBX 63
10.4 Instalación de Ruby + Rails 64

11. CONCLUSIONES Y MEJORAS 65
11.1 Grabación de llamadas 65
11.2 Voicemail 65
11.3 Fax2mail 65
11.4 Mail2fax 65
11.5 Llamadas vía WebRTC 65
11.6 Facturación 65

11.7 DOMÓTICA 66
11.8 PLANES DE MARCADO INTELIGENTE PARA CALL-CENTERS 66
12. REFERENCIAS Y BIBLIOGRAFÍA 67
12.1 PROYECTO 67
12.2 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS 67
12.3 MATERIAL DE APOYO 67

Índice de ilustraciones

Ilustración 1 – Configuración de una extensión en FreePBX 12
Ilustración 2 – Panel de configuración de llamada en Elastix 13
Ilustración 3 –Logotipo de Asterisk 17
Ilustración 4 –Traza de comunicación del protocolo SIP 18
Ilustración 5 –Ejemplo de orden de timbrado en el fichero extensions.conf 19
Ilustración 6 – Teléfonos IP de la marca Grandstream 20
Ilustración 7 – Logotipo de Ruby on Rails 21
Ilustración 8 – Logotipo de PHP 22
Ilustración 9 – Logotipo de MySQL 22
Ilustración 10 – Tablero Trello 24
Ilustración 11 – Tablero Trello con el desarrollo iniciado 24
Ilustración 12 – Tablero Trello con el desarrollo avanzado 25
Ilustración 13 – Tablero Trello con el desarrollo finalizado 25
Ilustración 14 – Tarjeta del tablero Trello del ticket “Llamadas entrantes, externas e internas” ... 26
Ilustración 15 – Logotipo de Github 28
Ilustración 16 – Logotipo de Travis CI 28
Ilustración 17 – Github configurado mediante Travis CI. Los tests han dado OK. 29
Ilustración 18 – Vista de un PR desde la interfaz de Travis CI 30
Ilustración 19 – Vista de Extensiones telefónicas 32
Ilustración 20 – Vista de Departamentos 33
Ilustración 21 – Vista de Numeración interna 34
Ilustración 22 – Vista de Numeración entrante 35
Ilustración 23 – Vista de Numeración saliente 35
Ilustración 24 – Vista de Locuciones 36
Ilustración 25 – Vista de Horarios 36
Ilustración 26 – Vista de Dias festivos 37

Ilustración 27 – Vista de Menú selectivo 38
Ilustración 28 – Vista de Menú selectivo al seleccionar un menú en concreto 38
Ilustración 29 – Vista de Estadísticas 39
Ilustración 30 – Filtro de fecha 39
Ilustración 31 – Resultado filtrando por fecha 40
Ilustración 32 – Filtro por departamento 40
Ilustración 33 – Resultado con ambos filtros 40
Ilustración 34 – Vista de Estadísticas, parte de facturación 41
Ilustración 35 – Vista de Trabajadores activos 42
Ilustración 36 – Ejemplo de fichero *.call 42
Ilustración 37 – Desvío de llamada 43
Ilustración 38 – Vista de Mini CRM 43
Ilustración 39 – Vista general de Mini CRM 44
Ilustración 40 – Diagrama del modelo relacional 46
Ilustración 41 – Métodos que atacan al API de Rails 47
Ilustración 42 – Ejemplo de la tabla extensions 48
Ilustración 43 – Menú de configuración 50
Ilustración 44 – Vista de Horarios 50
Ilustración 45 – Vista de Horarios seleccionando un horario en concreto 51
Ilustración 46 – Vista de departamentos mostrando opciones de horario y extensiones 51
Ilustración 47 – Vista de Extensiones telefónicas 52
Ilustración 48 – Tabla Horario 52
Ilustración 49 – Tabla HorarioLocucion 52
Ilustración 50 – Tabla Audio 53
Ilustración 51 – Tabla Extension 53
Ilustración 52 – Vista de Menú selectivo 54
Ilustración 53 – Vista de Menú selectivo seleccionando un menú en concreto 54

Ilustración 54 – Tabla Menu 54

Ilustración 55 – Tabla MenuOption 55

Ilustración 56 – Tabla Extensions 55

Ilustración 57 – Métodos que atacan al API de Rails de la capa Asterisk 56

Ilustración 58 – Test unitario 1 61

Ilustración 59 – Test unitario 2 61

Ilustración 60 – Test unitario 3 62

Ilustración 61 – CLI de Asterisk 64

1. Introducción

Este proyecto trata de la realización de un software que permita la instalación e integración de una centralita virtual en cualquier tipo de empresa o vivienda particular. Todo esto de manera sencilla y rápida, solo teniendo unas nociones muy básicas de VoIP y pudiendo realizar todo tipo de configuraciones básicas y avanzadas para cualquier compañía.

La aplicación a desarrollar permitirá entre otras funcionalidades la configuración de extensiones telefónicas, departamentos, horarios, locuciones de bienvenida o menús. También tendrá un diseño y arquitectura que la hará fácilmente extensible para incluir un sinfín de posibilidades más.

2. Estado del arte

En esta sección vamos a revisar las aplicaciones más populares existentes que se hallan en el mercado con el fin de tener un concepto general sobre el software que se va a enfocar.

2.1 FreePBX

Este software es la interfaz gráfica para Asterisk[1], que permite configurar de forma sencilla y muy básica una centralita virtual, así como también la creación de usuarios, extensiones, ordenes de timbrado, etc. A su vez, está implementado utilizando PHP[2] y MySQL[3]. No obstante, para un mayor control es necesario recurrir a la línea de comandos para realizar configuraciones, aunque no todos los módulos de Asterisk están soportados.

Por otro lado, muestra datos muy técnicos dirigidos a un personal informático convencional.

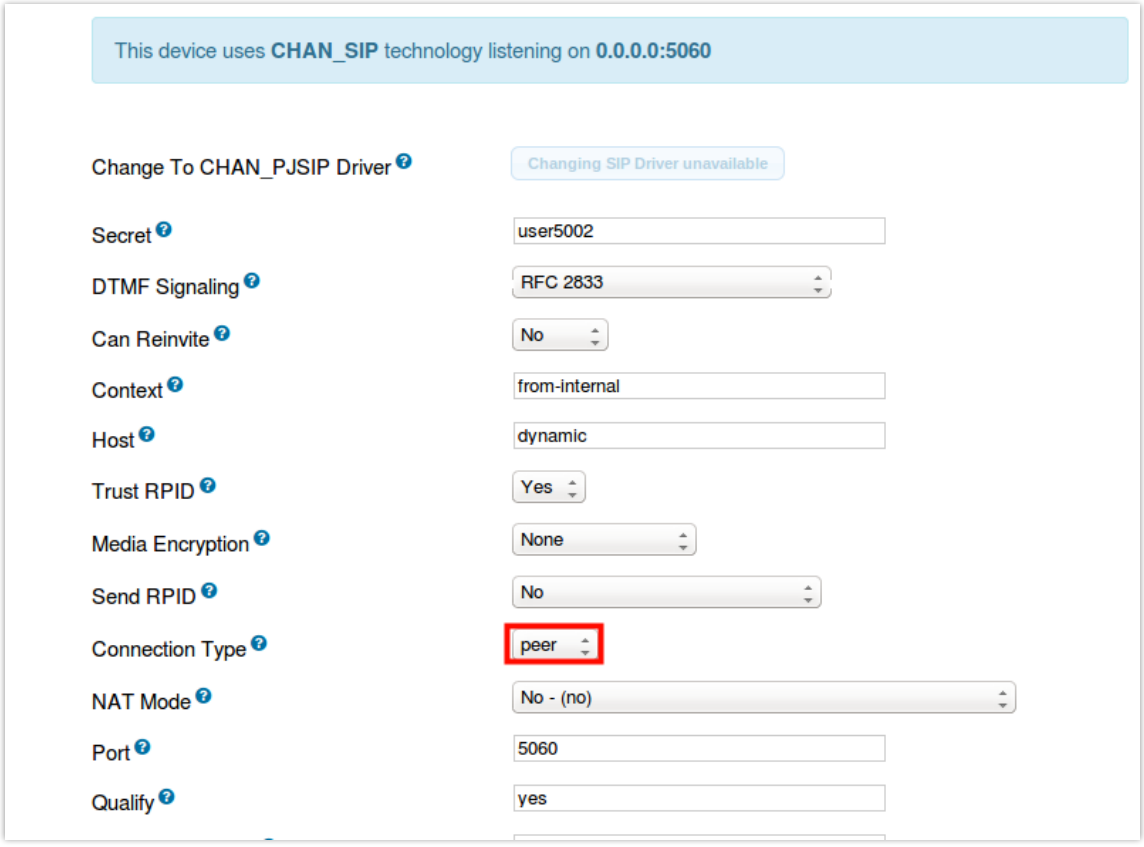

Ilustración 1 – Configuración de una extensión en FreePBX

[1] Asterisk – Software de código abierto programado en lenguaje C basado en una centralita telefónica
[2] PHP – Lenguaje de programación
[3] MySQL – Gestor de bases de datos relacionales

2.2 Elastix

Es una distribución open-source[4] creada en América Latina, que utiliza la interfaz de FreePBX y está enfocada a tarificación de llamadas y, por tanto, a revendedores de minutos de marca blanca (operadores no oficiales de la CMT[5]).

Elastix no reemplaza Asterisk. Se trata de un conjunto de herramientas unidas que nos facilitan las labores más comunes que haríamos con un Asterisk puro, no obstante, instala muchos componentes por defecto, aunque no se quieran usar y hace la interfaz lenta y pesada, en comparación con FreePBX puro. Tener muchos complementos instalados conlleva a errores de seguridad por falta de actualización de los mismos.

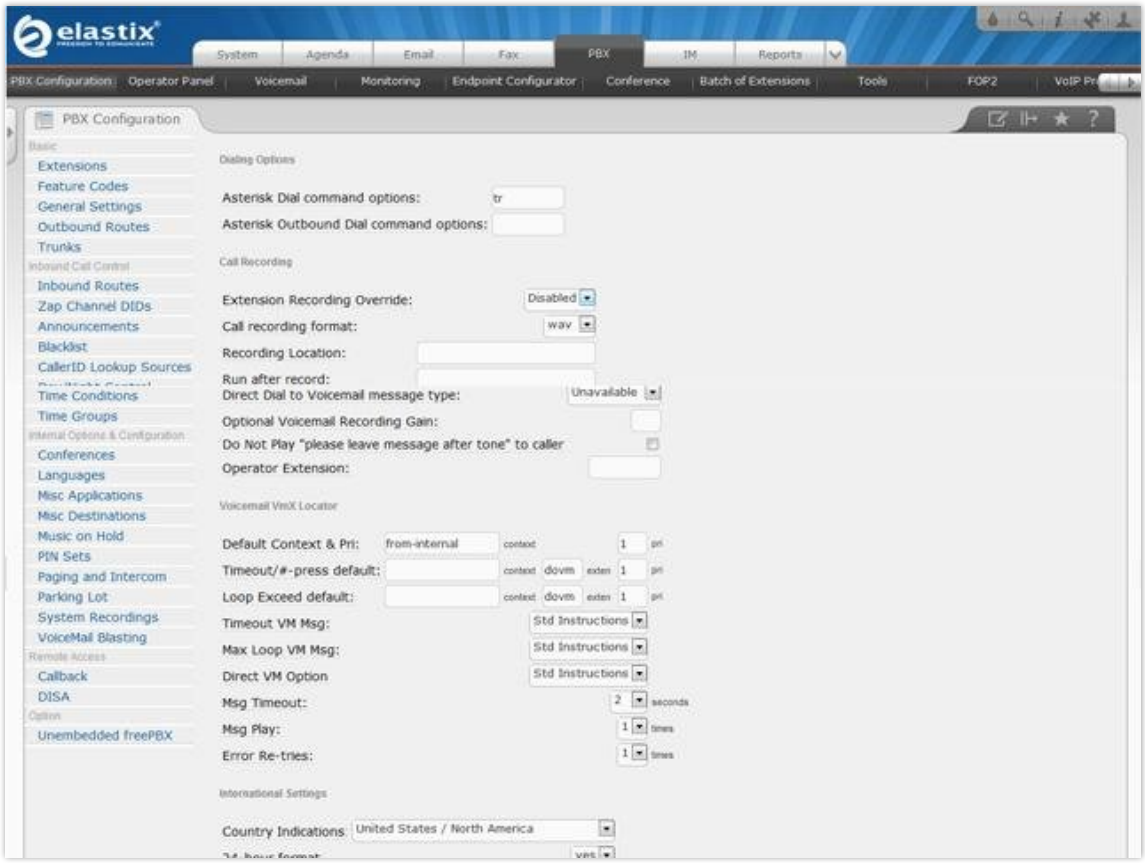

Ilustración 2 – Panel de configuración de llamada en Elastix

2.3 Trixbox

Es otra distribución que hace uso de FreePBX, MySQL, PHP, CentOS[6] y Asterisk, siendo la versión más usada en el mercado norteamericano. Esta utiliza la misma interfaz que FreePBX, la cual parte

[4] Open-source – Tipo de software de código abierto
[5] CMT – Comisión del Mercado de las Telecomunicaciones de España
[6] CentOS – Sistema operativo de tipo Linux

de un fork de una versión antigua de FreePBX. Se puede administrar en su versión PRO desde la nube, pero sus componentes son muy viejos, no tiene soporte para el mercado de América Latina y la plataforma tiene poco desarrollo.

2.4 AsteriskNow

Es la distribución oficial de Digium[7]. Es la más ligera de todas, ya que no se instalan extras. El FreePBX viene puro, por lo que se puede utilizar la versión más reciente. También es la distribución que más rápidamente ofrece las nuevas actualizaciones para Asterisk.

Al ser mantenida por Digium, no ofrece el soporte para tarjetas PSTN[8] de sus competidores. En caso de necesitar estos drivers, es necesario instalarlos aparte y lo peor de todo, mediante el terminal.

2.5 Conclusiones

Las diversas distribuciones que he encontrado con un propósito similar a mi propuesta de proyecto requieren un conocimiento medio/alto del personal informático que se encargue de la administración de una centralita virtual para una empresa, ya sea pequeña, mediana o grande.

En comparación con otras aplicaciones, encuentro que algunas de estas cubren en mayor medida los propósitos de mi proyecto, sin embargo, no tienen en consideración algunos puntos de vital importancia, como por ejemplo, no depender de la línea de comandos para realizar alguna operación, tener una interfaz clara y sencilla, no requerir conocimiento alguno sobre VoIP, o instalar solamente el software requerido por las funcionalidades que aporte este. Es por esto, que mi propuesta sí que incorporará dichos elementos, lo que hará que la aplicación resultante sea más liviana.

Finalmente, en caso de realizar alguna ampliación con más funcionalidades y de necesitar cualquier módulo de Asterisk, se podrá añadir de forma sencilla, ya que este proyecto utiliza Asterisk puro para poder exprimirlo al 100% sin ninguna limitación.

[7] Digium – Empresa de telecomunicaciones fundada por Mark Spencer
[8] PSTN – Red telefónica conmutada

	FreePBX	Elastix	Trixbox	AsteriskNow	Mi propuesta
Se puede prescindir del terminal	✗	✓	✓	✗	✓
Soporta todos los módulos de Asterisk	✗	✓	✓	✓	✓
Ahorra la instalación de componentes innecesarios	✗	✗	✗	✗	✓
Apto para personal no conocedor de VoIP	✗	✗	✗	✗	✓

3. Objetivos

3.1 Centralita Virtual (PBX)

El objetivo principal por el que realizo este proyecto es suplir la necesidad de que el usuario final requiera de un conocimiento específico en VoIP, esto es, que cualquier DevOps[9] de una empresa pueda realizar la configuración de una centralita virtual de manera sencilla y amena, sin tener que recurrir a extensos manuales ni pedir ayuda al proveedor de telefonía.

Por ello, la propuesta principal de este trabajo es la creación de un software que actúe de capa de abstracción sobre una centralita virtual a bajo nivel, utilizando un Asterisk base puro, sin ningún añadido adicional que haría aumentar notablemente la complejidad de su configuración donde la complejidad es notablemente alta a la hora de su configuración.

3.2 Administración

La idea del proyecto es que casi todo gire en torno a la configuración y administración de la centralita virtual, mediante un panel con las diversas funcionalidades donde se puedan crear extensiones telefónicas, agruparlas por departamentos, gestionar la numeración entrante y saliente, locuciones para el IVR[10] y un largo etc. Esto dependerá de hasta dónde queramos ampliar nuestro software, ya que el mundo de la VoIP da un sinfín de posibilidades.

[9] DevOps – Administrador de sistemas
[10] IVR – Respuesta de voz interactiva

4. Tecnologías

4.1 Asterisk

Asterisk es un software open-source creado por Mark Spencer, que fue lanzado en 1999. Está programado en lenguaje C y proporciona funcionalidades de una centralita telefónica o PBX, como por ejemplo: creación de buzones de voz, multiconferencias, IVR, distribución automática de llamadas, etc. Los usuarios pueden utilizar y crear todas las configuraciones que deseen e incluso crear alguna funcionalidad adicional, programando en el lenguaje de script de Asterisk.

Ilustración 3 –Logotipo de Asterisk

Además, Asterisk soporta varios protocolos de comunicación como son SIP, H.323, IAX y MGCP; utilizando el protocolo SIP en este trabajo, pues es el más extendido entre la comunidad VoIP.

Finalmente, un valor añadido de Asterisk es la posibilidad de interconectar y unificar distintas tecnologías: VoIP (Voz sobre IP), GSM (Comunicación móvil) y PSTN.

Daremos una breve introducción al extenso mundo de Asterisk, detallando unas definiciones básicas para poder entender un poco más a fondo cómo funciona el flujo de una llamada en la centralita.

4.1.1 Protocolo SIP

Se trata de un protocolo de iniciación de sesión que consiste en el intercambio de mensajes entre las partes que quieren comunicarse, para el establecimiento de sesiones multimedia entre dos o más usuarios. El funcionamiento es similar al protocolo HTTP, compartiendo algunos de los principios de diseño, como son la estructura de petición-respuesta y algunos códigos de estado (como por ejemplo el famoso 404 -not found-).

Asimismo, SIP no sólo se limita a la comunicación de voz, sino que también se extiende a comunicaciones de vídeo. Los promotores del protocolo SIP tienen sus raíces en la comunidad IP y no en la industria de las telecomunicaciones, por lo cual, este es otro de los motivos por los que se ha decantado la balanza al uso de dicho protocolo.

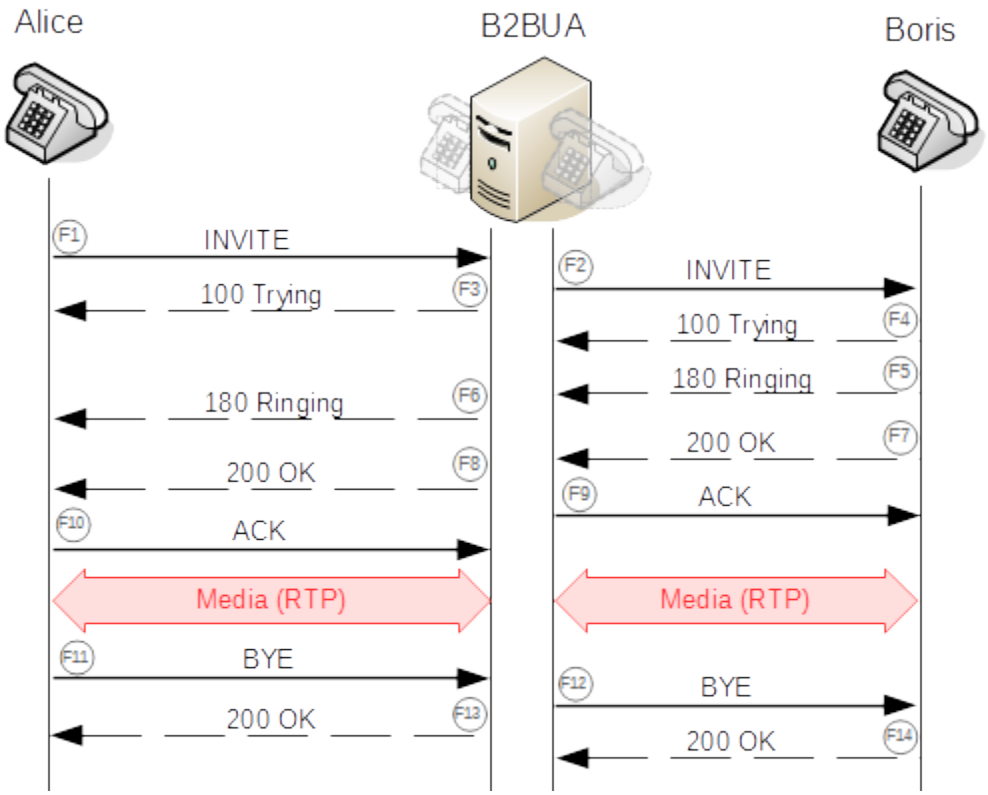

Ilustración 4 –Traza de comunicación del protocolo SIP

4.1.2 Protocolo RTP

El protocolo de transporte en tiempo real o RTP[11] es un protocolo de nivel de aplicación que se utiliza para transmitir información en tiempo real; en concreto Asterisk lo usa para la transmisión de voz. Existen diferentes versiones, como la versión con control de protocolo o RTCP[12], y la versión segura o SRTP[13]. Asimismo, dichas versiones pueden estar combinadas para proporcionar mayor seguridad en la comunicación o SRTCP[14].

4.1.3 Dialplan

En Asterisk, llamamos dialplan a la configuración del flujo de llamadas que pasan por la centralita. Está escrito en lenguaje script de Asterisk y se halla concretamente en el fichero **/etc/Asterisk/extensions.conf**. A modo de ejemplo, una breve explicación de un dialplan sencillo sería la siguiente:

[11] RTP – Real-time Transport Protocol
[12] RTCP – Real-time Control Protocol
[13] SRTP – Secure Real-time Transport Protocol
[14] SRTCP – Secure Real-time Transport Control Protocol

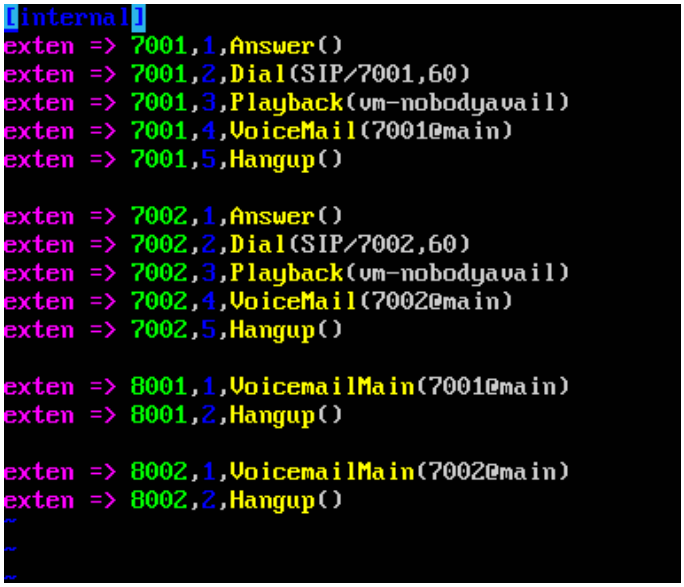

Ilustración 5 –Ejemplo de orden de timbrado en el fichero extensions.conf

Tomemos la segunda instrucción como ejemplo, Exten => 7001, 2, Dial(SIP/7001,60), que contiene los siguientes datos:

- **Exten =>**: Comienzo de línea de instrucción del dialplan.
- **7001**: Corresponde al número de extensión.
- **2**: Corresponde al orden de prioridad, ya que se leen en orden secuencial.
- **Dial(SIP/7002,60)**: Instrucción que realiza una llamada a la extensión 7002 mediante un canal de tipo SIP, durante 60 segundos.
- **Dial(SIP/7002,60)**: Instrucción que realiza una llamada a la extensión 7002 mediante un canal de tipo SIP, durante 60 segundos.
- **Dial(SIP/7002,60)**: Instrucción que realiza una llamada a la extensión 7002 mediante un canal de tipo SIP, durante 60 segundos.

Como vemos, Asterisk posee su propio lenguaje script para manejar el flujo de llamadas.

4.1.3.1 ¿Qué significa dialplan configurado en Real Time?

La configuración del orden de timbrado se realiza directamente en el fichero extensions.conf, con el inconveniente de realizar una instrucción en el *Command Line Interface* (CLI) de Asterisk cada vez que se modifique dicho fichero. La instrucción que se debe ejecutar en el CLI es “dialplan reload”, la cual hace que Asterisk vuelva a recargar la configuración del fichero y poder trabajar con los parámetros nuevos.

Asterisk nos proporciona un módulo que podemos instalar y configurar para que dicha configuración del fichero extensions.conf se plasme en una base de datos, ya que tiene un campo de prioridad. Como hemos visto en el apartado Dialplan, la configuración en Real Time permite que

aunque se desordenen las filas en base de datos, se mantenga el orden secuencial que Asterisk tendría en dicho fichero.

4.1.3.2 ¿Cómo conecto un teléfono para llamar mediante Asterisk?

Para ello necesitamos un terminal IP, que a fin de cuentas es un teléfono con una conexión RJ45 (cable de red) en vez de RJ11 (cable de teléfono tradicional).

Al darle corriente al teléfono y conectarlo a un router, gracias a la obtención de dirección IP mediante DHCP[15] del teléfono, se le asignará una dirección IP. Para configurar el teléfono con las credenciales, solamente tenemos que introducir dicha IP en un navegador de un computador que se halle en la misma red que el teléfono para poder entrar a la página de su configuración.

Dependerá de cada modelo y fabricante el diseño de la página de configuración. Los fabricantes más famosos de telefonía IP son Cisco, Grandstream, Snom, Alcatel, Panasonic, entre otros.

Ilustración 6 – Teléfonos IP de la marca Grandstream

4.1.3.3 ¿Qué es un Trunk SIP?

Antes de empezar a detallar el funcionamiento del flujo de llamadas, necesitamos saber unas nociones básicas, como el concepto de Trunk SIP.

Un Trunk SIP es una configuración que nos permite conectarnos a otro sistema o centralita, ya sea analógica o digital. Lo habitual es que cuando se contratan servicios de VoIP con un operador, este nos proporciona una centralita virtual en la nube (y la gestión corre a cargo del operador según nuestro criterio), o nos proporciona una conexión vía Trunk. En este caso, conectaríamos el Trunk SIP en nuestra centralita y ya tendríamos conexión para sacar o recibir llamadas, dependiendo de lo que hayamos contratado.

[15] DHCP – Protocolo de red de tipo cliente/servidor para la asignación automática de direcciones IP

4.1.3.4 ¿Cómo funciona el flujo de llamadas en Asterisk?

Supongamos que tenemos un Trunk SIP habilitado para recibir y emitir llamadas, así como también un número geográfico contratado y configurado en la centralita (por ejemplo, el 965 123 456), y alguien realiza una llamada desde un teléfono móvil convencional a nuestro número de la centralita (965 123 456). De esta forma, los pasos del flujo son:

1. Nuestro operador recibe la llamada y nos la reenvía mediante el Trunk SIP que tenemos contratado y configurado con el operador.
2. En la configuración del Trunk SIP (fichero /etc/Asterisk/sip.conf o Real-time) reenviamos la llamada hacia nuestro dial plan, al grupo de extensión que se precise.
3. La llamada pasa por nuestra centralita y, al tenerla en nuestro poder, podemos hacer con ella lo que nos convenga hasta finalmente hacer que suene en un terminal, dos o los que sean oportunos (también podemos reenviar la llamada a otro teléfono fijo o móvil).
4. Al descolgar un terminal de nuestra centralita, estaremos atendiendo a esa llamada que se realizó a nuestro número geográfico (965 123 456). Una vez se finalice la comunicación y se cuelgue, el flujo habrá finalizado mediante la instrucción Hangup.

Para realizar una llamada saliente, simplemente deberemos encaminar dicha llamada hacia el TrunkSIP del operador para tener salida al exterior.

4.2 Ruby on Rails

La interfaz web de la aplicación se ha creado mediante el framework Ruby on Rails, ya que proporciona velocidad en su desarrollo gracias a la buena curva de aprendizaje que tiene y prácticamente a coste cero. Es una tecnología que proporciona buena reutilización de código y aumenta la productividad a la hora de programar con ella.

Ilustración 7 – Logotipo de Ruby on Rails

4.3 PHP

PHP es el lenguaje que se ha decidido utilizar en este trabajo para realizar toda la base de scripting que necesita la configuración interna de la centralita virtual a nivel de dialplan, con el fin de nutrirse de datos externos a esta. Aprovechando la librería PHPAgi hacemos que el código quede más limpio y, sobre todo, eficiente en comparación con el método tradicional, (procedural) ya que permite la reutilización del mismo mediante objetos.

Ilustración 8 – Logotipo de PHP

4.4 MySQL

Se utiliza este sistema de gestión de base de datos relacionales para almacenar de manera persistente los datos de las configuraciones de la centralita virtual, aprovechando así la librería que posee Asterisk para esta tecnología. De esta forma, se conectan las infraestructuras, tanto la básica del proyecto (almacenamiento de clientes, datos, administradores, etc.) como la de Asterisk, utilizando el módulo de configuración de la centralita en tiempo real (*realtime*) e interconectando el sistema complejo de capas internas de configuración de la aplicación.

Ilustración 9 – Logotipo de MySQL

5. Metodología

5.1 Scrum y Kanban

A pesar de realizar el proyecto una sola persona, se ha escogido la conocida metodología Scrum[16] combinándola con la metodología Kanban, simulando una serie de sprints y la repartición de trabajo de forma similar al de un equipo de desarrollo.

En este caso, las reuniones semanales se han hecho cada 2 o 3 semanas con el tutor, que consistían en una demo de las historias de usuario realizadas hasta la fecha, comentando los posibles problemas encontrados y discutiendo las posibles soluciones al respecto. Al final de cada reunión, se han ido detallando historias nuevas y sus correspondientes descomposiciones en tickets.

Además, durante todo el desarrollo del proyecto se ha contado con la ayuda de un tablero Kanban[17], pues el proceso es mucho más llevadero, sencillo y fácil de seguir a medida que van pasando los días. Asimismo, se ha tomado la regla de la metodología Kanban de no avanzar una tarea hasta que no esté completamente acabada o no empezar una tarea hasta que no se haya terminado la anterior. De esta forma se garantiza un orden en el proceso de desarrollo.

Más adelante entraremos en detalle en la herramienta utilizada como tablero Kanban.

5.2 Tablero: Trello

Con el apoyo de esta herramienta se han ido creando historias de usuario, y por cada una de ellas, el correspondiente desglose en tickets. A la vez que se iba avanzando en el proyecto se iban actualizando los tickets y las columnas, para tener un control más preciso sobre el estado del desarrollo.

Podemos observar el progreso del tablero Kanban a medida que avanzó el desarrollo, sus sprints de 2-3 semanas y las historias de usuario organizadas por colores y tickets. En las siguientes imágenes podemos ver unos ejemplos del avance del proyecto.

[16] Scrum – Metodología de desarrollo ágil

[17] Kanban – Método para gestionar el trabajo intelectual con énfasis de entrega just-in-time

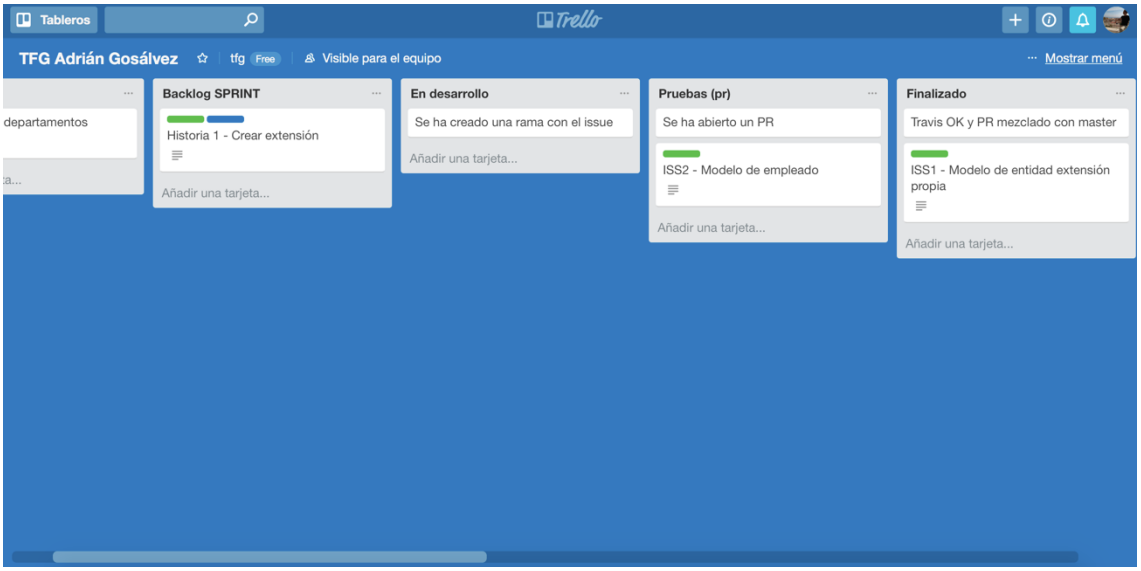

Ilustración 10 – Tablero Trello

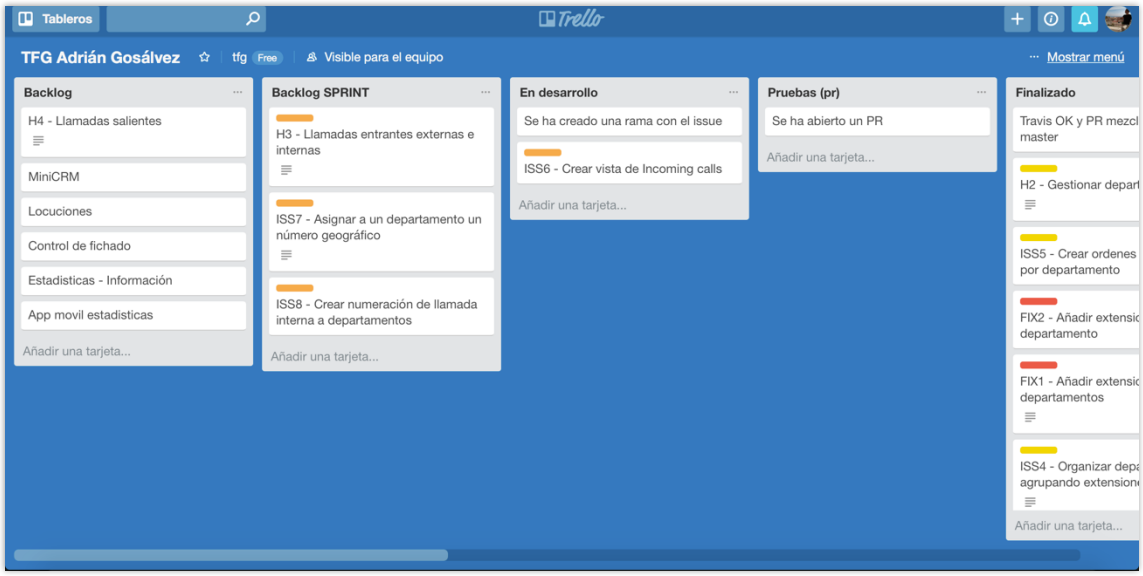

Ilustración 11 – Tablero Trello con el desarrollo iniciado

Aquí podemos observar varias columnas de sprints finalizados con las historias de usuarios y sus tickets correspondientes finalizados.

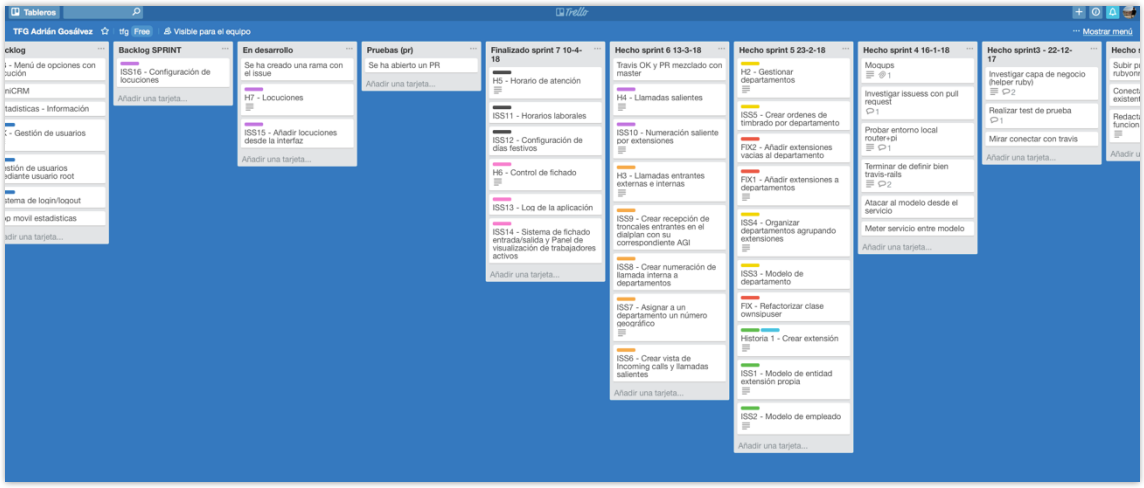

Ilustración 12 – Tablero Trello con el desarrollo avanzado

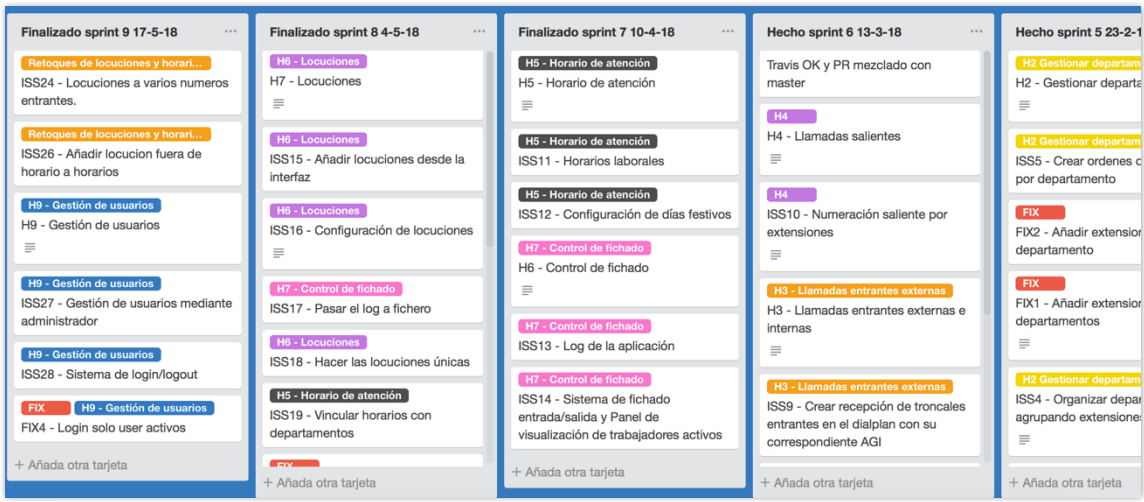

Ilustración 13 – Tablero Trello con el desarrollo finalizado

5.2.1. Columnas del tablero

Como en cualquier tablero Kanban, siguiendo la metodología Scrum, se han creado las siguientes columnas para agrupar las tarjetas de historias de usuario y tickets:

- Backlog. Se añadieron las historias de usuario pendientes para futuros sprints, así como ideas que iban surgiendo a medida del desarrollo.
- Backlog SPRINT (ToDo). Se añaden historias de usuario de la columna anterior y se desglosan en tickets para el sprint actual.
- En desarrollo (In progress). Conforme se iba avanzando en el desarrollo, se iban pasando tarjetas de la columna anterior a esta, mientras se estaba trabajando en dicha funcionalidad.

- Pruebas (Test + Integración). Una vez que se termina de trabajar en una funcionalidad, la pasamos a esta columna donde permanece mientras se realizan los test unitarios mediante el sistema de integración continua (de la cual hablaremos más adelante).
- Finalización Sprint (Done) – DD-MM-AA. Cuando una tarjeta se ha completado satisfactoriamente pasa a esta columna. Acto seguido se vuelve a escoger una tarjeta de la columna "Backlog Srpint", se pasa a desarrollo y se continúa desarrollando la nueva funcionalidad. Se ha creado una columna de este tipo por cada sprint finalizado.

Al ser una sola persona la que ha desarrollado dicho proyecto, se ha escogido un WIP (Work-in-prgress) de 1 por columna del tablero.

5.2.2. Tarjetas

Contienen la información, de las historias de usuario y tickets, necesaria para el desarrollo de la funcionalidad. Así pues, las historias de usuario se describen siguiendo un enunciado que se encuentra debajo del título de la tarjeta. El enunciado sigue el siguiente esquema: Como (tipo de usuario) quiero (conseguir algo) para (conseguir un objetivo). Posteriormente se escriben las condiciones de satisfacción de la funcionalidad. En la foto X se describe un ejemplo real del desarrollo de la aplicación.

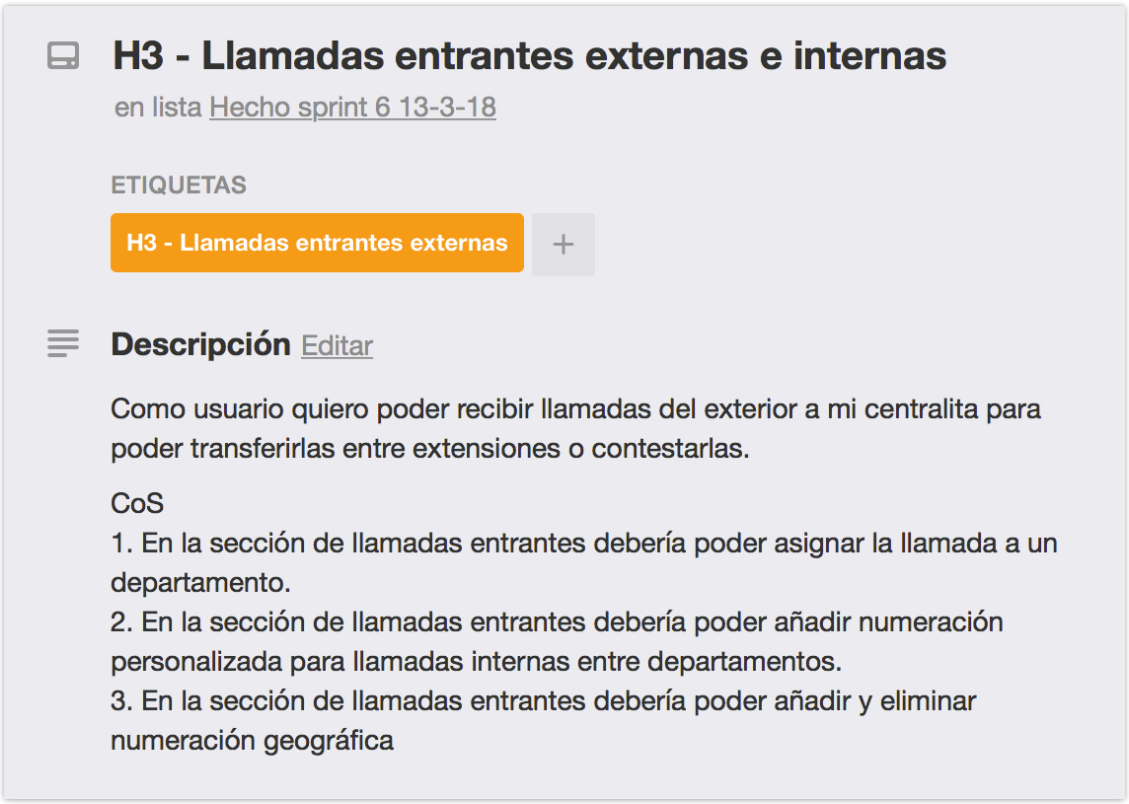

Ilustración 14 – Tarjeta del tablero Trello del ticket "Llamadas entrantes, externas e internas"

5.3 Flujo de trabajo: GitFlow

Se ha utilizado el control de versiones Git, y GitFlow[18] como modelo de trabajo, donde se pueden obtener una serie de beneficios para un desarrollo ágil, seguro y eficiente, ya que encaja bastante bien con las metodologías explicadas anteriormente, Scrum y Kanban.

Para cada ticket (es decir, una funcionalidad al realizar la descomposición de una historia de usuario), se ha procedido a realizar el flujo de trabajo denominado GitFlow, que consiste en crear una rama desde la rama Develop para trabajar en la funcionalidad. Al finalizar dicha rama, proceder con un Pull Request[19] (PR) hacia la rama develop y continuar con su correspondiente borrado. De esta forma se ha añadido cada uno de los tickets del proyecto a lo largo del desarrollo.

Las ramas de desarrollo que se han utilizado son las siguientes:

- Master: Contiene el código de producción.

- Develop: contiene el código que estaría en desarrollo.

- Release: contiene una versión para producción a partir de la rama Master.

- IssXX: se crea a partir de Develop, donde XX corresponde a la numeración de la historia de usuario correspondiente del tablero Kanban, y una vez finalizado el desarrollo, se integra con la rama Develop.

- FixXX: contiene código de corrección, donde XX se corresponde al número de la historia de usuario.

Imaginemos que se va a crear el primer ticket denominado “Crear usuarios”, que corresponde a la historia de usuario 3. Gestión de usuarios:

1.- Creamos la rama iss31 a partir de Develop

2.- Trabajamos en la rama

3.- Se hace PR a Develop

De esta forma todo queda organizado y se aíslan todas las funcionalidades, de modo que ante cualquier posible problema es muy sencillo localizarlo rápidamente mediante el control de versiones.

[18] GitFlow – Modelo de trabajo con ramas de desarrollo en Git
[19] Pull Request – Petición para mezclar código fuente entre ramas en un repositorio

5.4 Control de versiones e integración continua

Vamos a profundizar un poco sobre el trabajo realizado con el control de versiones y la Integración Continua (CI). En general, destacar que el repositorio se conectó a un sistema de CI para automatizar y agilizar el desarrollo.

5.4.1 GitHub

Durante todo el desarrollo se ha usado el sistema de control de versiones Git, mediante la plataforma Github, la cual destaca a nivel mundial; así como también ha sido comprada Microsoft recientemente. A través de esta, la metodología GitFlow tiene una aplicación sencilla, y además, al haber solamente un único desarrollador, no ha habido ningún conflicto entre ramas.

Ilustración 15 – Logotipo de Github

5.4.2 Travis CI

Travis es una plataforma de integración continua gratuita que nos permite conectar un repositorio, en este caso de Github, a un flujo de desarrollo mediante la configuración necesaria en el archivo .travis.yml, que se encuentra en la raíz del proyecto.

Ilustración 16 – Logotipo de Travis CI

Travisi CI es bastante flexible y ofrece gran variedad de opciones. En este proyecto, se ha configurado de forma que nada más realizar un Push o un Pull Request (PR) de la rama en la que se está trabajando se lance automáticamente, realice un despliegue del sistema (Ruby on Rails sobre MySql), inicialice la base de datos con una semilla (fichero seeds.rb o ficheros con extensión .yml) y lance los test unitarios de cada ticket. Al finalizar, Travis CI nos indica en el propio PR si ha pasado los test de manera satisfactoria o si por el contrario ha habido algún fallo en los mismos.

Desde la web de Github vemos la siguiente imagen, en la que Travis CI avisa que los test unitarios han sido ejecutados sin ningún problema y da luz verde para realizar el Merge[20] a la rama Develop.

Ilustración 17 – Github configurado mediante Travis CI. Los tests han dado OK.

En la siguiente imagen se observan las pruebas realizadas en Travis CI desde su página. De esta manera vemos que nos da luz verde y una serie de detalles de las pruebas, así como el tiempo de ejecución, el identificador del commit[21] en Github, el nombre del PR y hacia qué rama se realiza dicho PR.

[20] Merge – Mezcla de dos ramas de un repositorio
[21] Commit – Confirmación de un conjunto de cambios en un fichero de programación

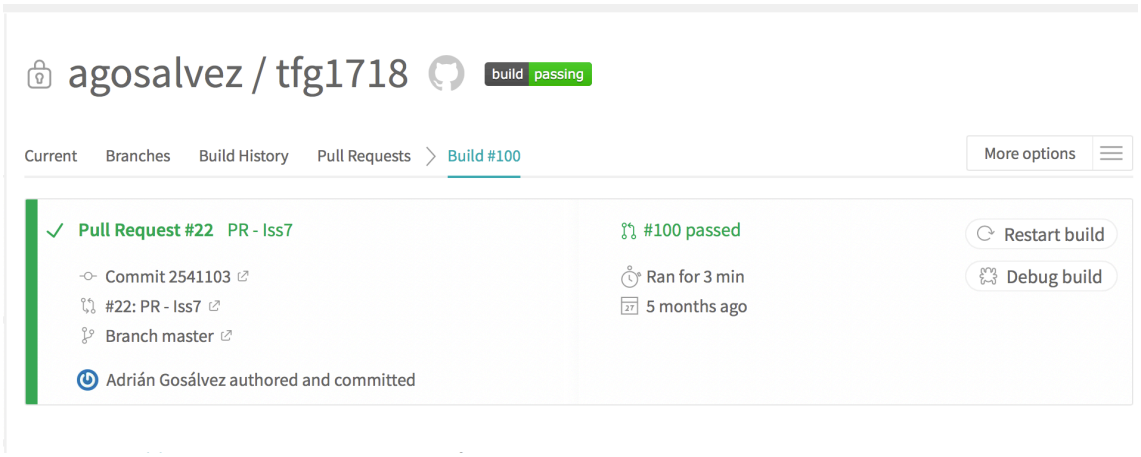

Ilustración 18 – Vista de un PR desde la interfaz de Travis CI

6. Funcionalidades

En este apartado, se definirán cada una de las funcionalidades que tiene la aplicación. Principalmente tenemos dos tipos de roles: el trabajador y el administrador del sistema.

Estas funcionalidades han sido desarrolladas de forma original, sin copiar ninguna de las aplicaciones existes, intentando obtener una aplicación con las características básicas necesarias para dar soporte a las necesidades principales de una empresa. Como se ha comentado en el apartado de metodología, para el diseño de las funcionalidades hemos utilizado una metodología iterativa. Al final de cada iteración se tenía una aplicación funcional con las nuevas características añadidas y se estudiaban y decidían las nuevas funcionalidades a desarrollar.

6.1 Rol Administrador del sistema

Denominamos de esta forma al usuario que tiene acceso total a la administración del software. Además de las opciones accesibles para el rol trabajador, también tiene todo el acceso a la configuración y parametrización de la red telefónica de la empresa: teléfonos, departamentos, gestión y control de llamadas, etc., las cuales vamos a detallar a continuación.

6.1.1 Gestión de usuarios

Consiste en un *Create-Retrieve-Update-Delete* (CRUD) para la gestión y administración de los usuarios del sistema. Este usuario se relacionará directamente con una extensión telefónica en el momento posterior a su creación, lo cual veremos más adelante.

6.1.2 Crear extensiones

Una de las funcionalidades principales es la de dar de alta una extensión telefónica. En la pantalla Extensiones telefónicas podemos ver las siguientes columnas: el número de extensión, la contraseña de configuración en un terminal IP, el empleado mediante el cual relacionamos dicha extensión telefónica, el departamento al que pertenece, si se encuentra activa en la centralita y una opción para eliminar la extensión.

Ilustración 19 – Vista de Extensiones telefónicas

¿Por qué se encuentra la contraseña en plano y directamente en la interfaz web?

Porque es necesario ver la contraseña asociada a cada extensión para poder realizar la configuración del terminal IP al darla de alta en el sistema, ya que los teléfonos tienen una página de configuración que se accede mediante su dirección local, y dicha sección solamente es visible para el administrador de la empresa, que es la persona que realizaría esta labor.

6.1.3 Crear departamentos

Muy vinculada a la sección anterior, tenemos la sección de creación y configuración de departamentos, donde podremos dar de alta departamentos en la empresa para agrupar extensiones telefónicas, ya que a fin de cuentas es un puesto de trabajo.

Para dar de alta un departamento solamente deberemos indicar el nombre de este. Seguidamente se añadirá a la lista con la configuración por defecto, donde ya podremos modificar tan sólo a golpe de clic la configuración de este.

La primera opción de configuración que encontramos es la de modo de timbrado, que nos permite decidir si los teléfonos de dicho departamento sonarán de forma consecutiva (según el orden en el que se añaden en la columna “extensiones”, la cual permite añadir las extensiones que necesitemos a dicho departamento, tiene relación N:N) o de manera simultánea al unísono.

Seguidamente tenemos el tiempo de timbrado, que nos permite decidir cuánto tiempo queremos que esté la llamada del cliente sonando en el departamento.

Por último, tenemos la opción de añadir un horario de atención telefónica a un departamento en concreto, ya que no suelen tener el mismo horario un departamento comercial, uno de desarrollo o uno de resolución de incidencias 24h. Esta sección se explicará más delante de forma detallada.

Ilustración 20 – Vista de Departamentos

6.1.4 Numeración

En esta sección vamos a controlar configuración relacionada directamente con alias personalizados de números para poder llamar directamente a extensiones y departamentos, permisos de llamada para empleados y numeración entrante.

6.1.4.1 Interna

Esta sección sirve principalmente para asignar una numeración personalizada a una extensión concreta o departamento.

Si queremos llamar directamente al departamento técnico y nos da igual la persona que nos atienda, le asignamos una numeración personalizada a dicho departamento. Por ejemplo, al departamento técnico se le asigna la numeración personalizada “1200”, de manera que al llamar desde un terminal a dicho número, sonará el departamento asignado.

Llegados a este punto, nos encontramos la interfaz dividida en dos. Por un lado, en la derecha se halla lo descrito anteriormente, y en la parte izquierda encontramos la misma funcionalidad, pero aplicada a extensiones concretas. Esto se ha puesto de modo extra, ya que por defecto, si una extensión tiene la numeración 101, al llamar a este número desde cualquier teléfono, sonará dicha extensión. Asimismo, existe la posibilidad que se implante el software en una empresa donde ya tenía unas numeraciones asignadas previamente y por comodidad se desea mantener dicha

configuración; en este caso esta funcionalidad permite que se mantengan los números previamente.

Ilustración 21 – Vista de Numeración interna

6.1.4.2 Entrante

Esta sección es una de las más importantes para el funcionamiento de la centralita a nivel externo. Aquí tiene lugar la configuración de la numeración entrante, es decir, las llamadas recibidas a un número geográfico (número fijo convencional) o número internacional que hayamos contratado previamente con un operador de telefonía IP.

Una vez que el operador confirma la redirección del tráfico hacia nuestro Trunk SIP, que tendremos registrado en su servidor, solamente tendremos que añadir dicho número contratado y elegir cómo redirigirlo dentro de nuestra centralita. También podemos asignar una locución, un menú IVR y a qué departamento irá dirigido.

La configuración del Trunk SIP se realizará al contratar los servicios con el operador de VoIP, siguiendo las instrucciones que nos proporcione, ya que es un hecho que varía dependiendo de la forma de trabajar de cada operador.

Ilustración 22 – Vista de Numeración entrante

6.1.4.3 Saliente

Otra funcionalidad útil es la de restringir llamadas. Para ello se ha diseñado una interfaz con el objetivo de limitar las llamadas por extensión, aunque también se podría realizar una limitación por departamento o incluso mixta, pues tenemos un sinfín de posibilidades debido a la flexibilidad que nos aporta el sistema tal como está planteado.

Principalmente se han separado las llamadas por tipo, como son las llamadas a números nacionales, numeración móvil, números internacionales y números de red inteligente (números que comienzan por 90X, 80X y su familia de tarificación adicional, a excepción de los 900 y 800, que son gratuitos).

Ilustración 23 – Vista de Numeración saliente

6.1.5 Locuciones

Todos hemos escuchado alguna vez las locuciones de bienvenida cuando llamamos a alguna empresa y nos mantienen en espera con publicidad o con información útil de la empresa como por

ejemplo, el horario de atención al público o un mensaje comunicando que se encuentran todas las líneas ocupadas y que lo intentemos más tarde. De todo esto trata esta funcionalidad.

En la interfaz de esta sección podemos añadir archivos de audio para utilizarlos en las secciones de menú IVR y horarios.

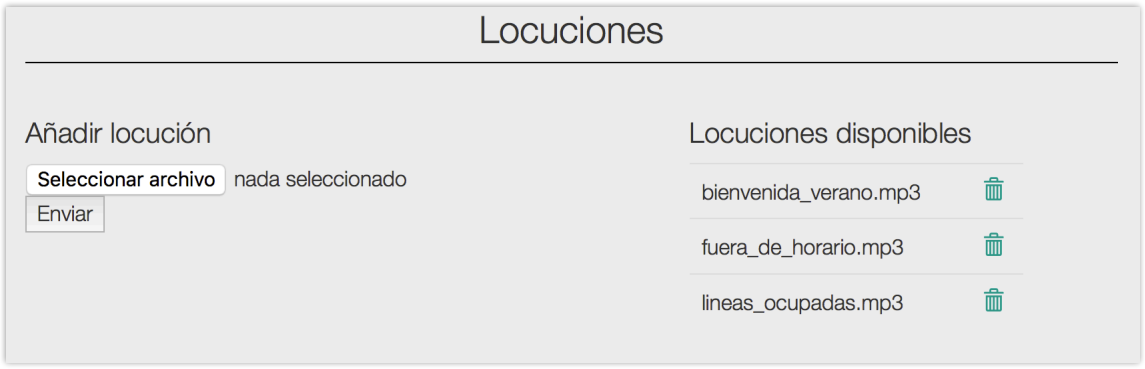

Ilustración 24 – Vista de Locuciones

6.1.6 Horarios

También podemos definir una serie de horarios dependiendo de la jornada laboral de la empresa, y como hemos nombrado en el apartado anterior, añadir una locución fuera de horario, que sonará cuando se realice una llamada fuera del horario indicado.

Ilustración 25 – Vista de Horarios

Además de indicar horarios, se ha añadido una subsección dentro de esta interfaz para establecer días festivos. Esto resulta muy interesante, pues cada comunidad autónoma o ciudad cuenta con días festivos diferentes, dando la oportunidad de establecer dichos días como "fuera de horario".

Ilustración 26 – Vista de Dias festivos

6.1.7 Menú de llamada (IVR)

Igual que las locuciones, también hemos pasado por un menú selectivo (IVR) al llamar a alguna compañía telefónica, que nos va redirigiendo para hablar con un departamento u otro en función de lo que necesitemos, después de indicárselo por teclado o mediante reconocimiento de voz.

En la sección de menú podemos añadir hasta 5 opciones de manera predeterminada, indicando el dígito a pulsar y escogiendo hacía donde se va a redirigir la llamada, que será hacia un departamento en sí o hacia una extensión concreta. Una vez hemos adjuntado las locuciones en el apartado correspondiente, tendremos disponible la elección de dichas locuciones. En el caso de la imagen siguiente, tendremos que elegir una locución de bienvenida que sonará junto al menú IVR. Normalmente estas locuciones añaden la información que se le da al cliente para que pulse una determinada tecla en función de lo que necesite, y la centralita leerá mediante detección DTMF la señal marcada por el cliente y finalmente, redirigirá la llamada.

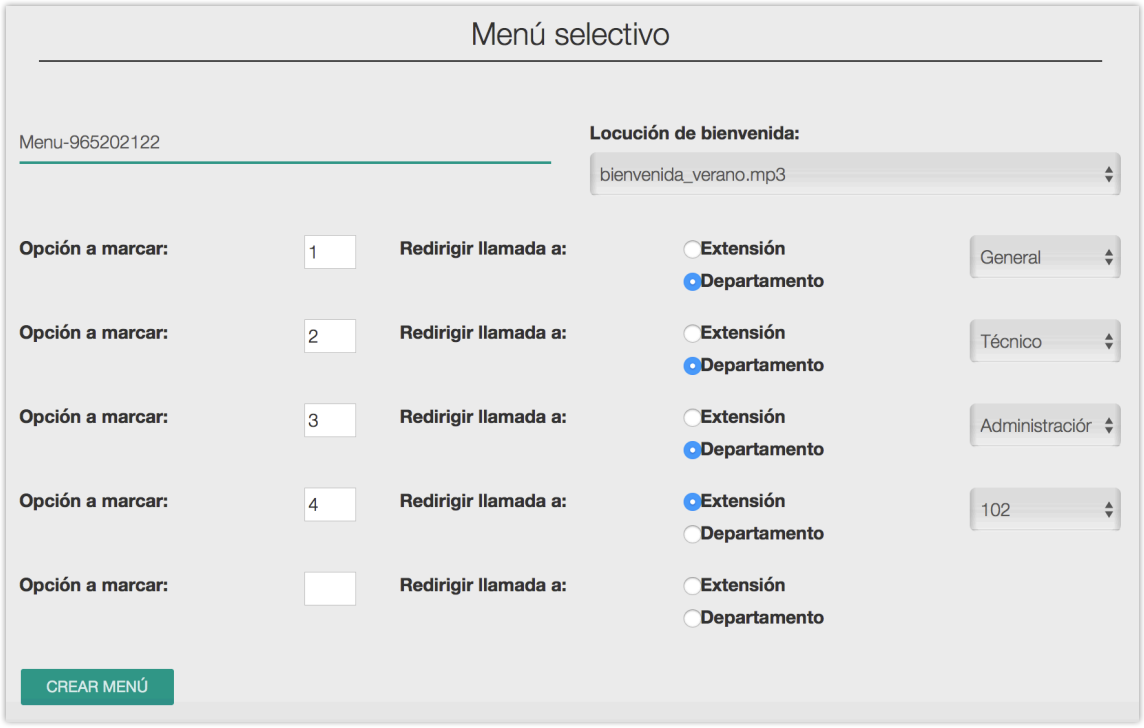

Ilustración 27 – Vista de Menú selectivo

Ilustración 28 – Vista de Menú selectivo al seleccionar un menú en concreto

6.1.8 Estadísticas

Por último, tenemos una sección dedicada a estadísticas sobre las llamadas realizadas en la centralita de telefonía. Es una mera indicación del potencial que tenemos para explotar los datos.

Se han expuesto dos gráficas indicando la cantidad de minutos consumidos agrupados por extensión en la parte izquierda, y en la derecha, la cantidad de llamadas agrupadas por extensión.

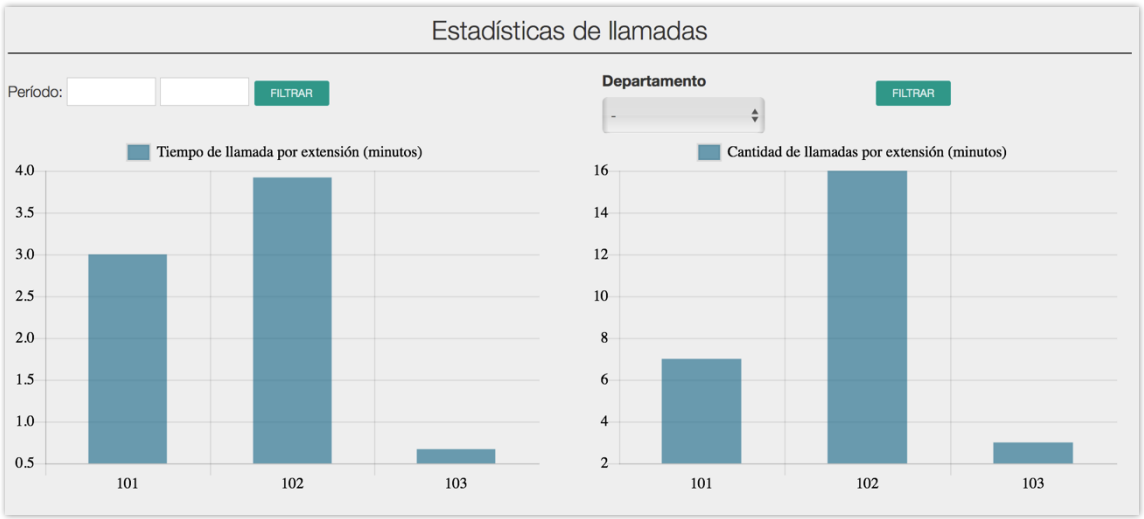

Ilustración 29 – Vista de Estadísticas

Además, en la parte superior tenemos filtros para agudizar la búsqueda, filtrando por rango de fechas y por departamento.

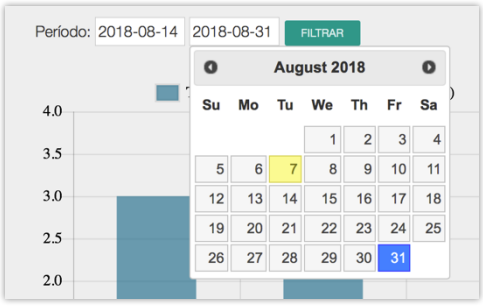

Ilustración 30 – Filtro de fecha

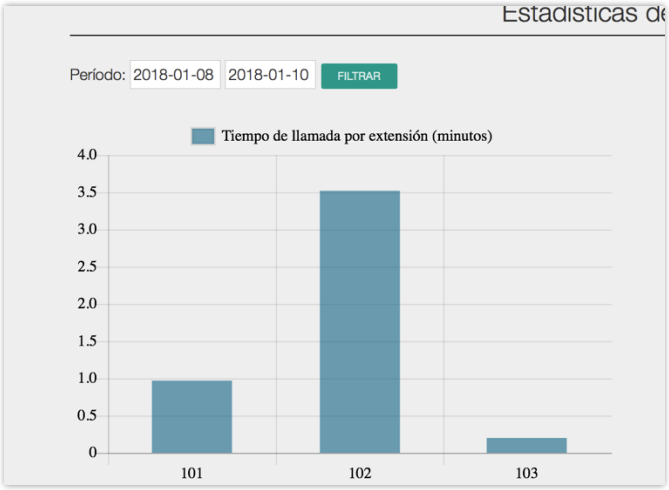

Ilustración 31 – Resultado filtrando por fecha

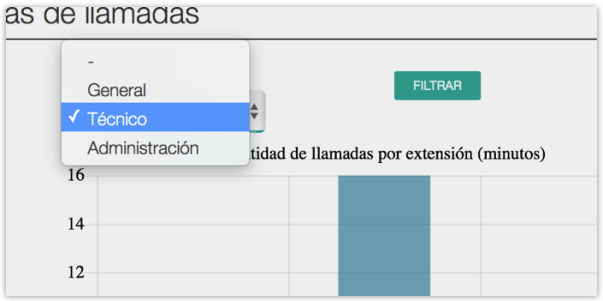

Ilustración 32 – Filtro por departamento

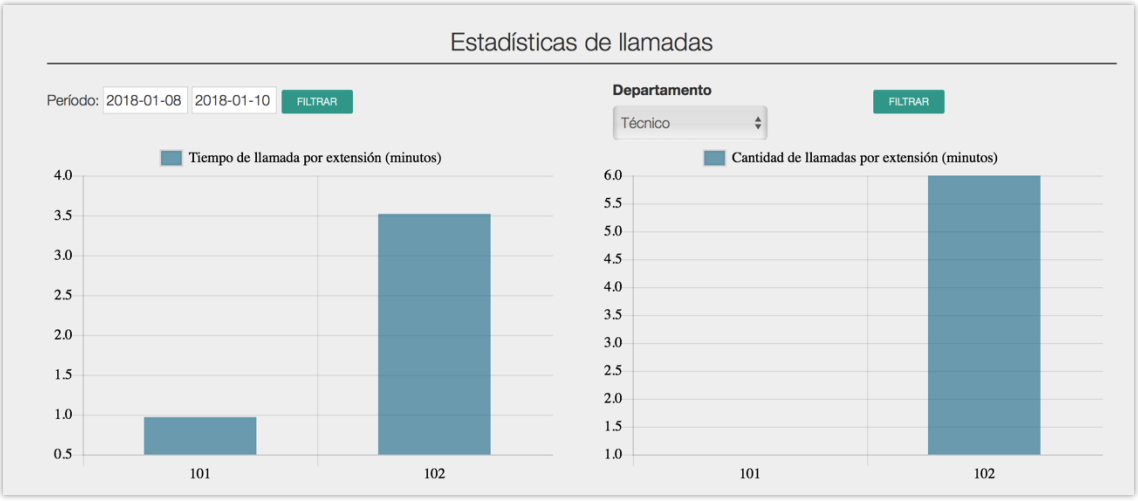

Ilustración 33 – Resultado con ambos filtros

También se proporcionan datos acerca de los consumos de la centralita, el tipo de factura del proveedor IP, los precios del consumo y un sinfín de posibilidades que podemos añadir.

Ilustración 34 – Vista de Estadísticas, parte de facturación

Para todo esto nos nutrimos de los datos almacenados en el *Call Detail Record* (CDR), que es un registro de todas las llamadas que se realizan en la centralita. El CDR también se ha pasado a *Realtime* y por consiguiente, se obtienen los datos de forma dinámica y directa mediante una API REST[22] hecha en Ruby On Rails.

6.2 Rol Trabajador

Llamamos así al trabajador de la empresa totalmente ajeno a las labores informáticas y de configuración de la telefonía (personal de administración, comerciales, etc.). De momento se ha decidido que solamente tendrán acceso a tres funcionalidades relacionadas con la centralita virtual y el teléfono IP de su puesto de trabajo, que detallamos a continuación.

6.2.1 Control de fichado

Se ha implementado un control de fichado de trabajadores (la sección del menú está indicada como "Personal"). Mediante una simple llamada a una combinación de números (que establecería el administrador del sistema en el momento de la instalación) desde el teléfono del propio puesto de trabajo de un empleado, se puede fichar y cerrar el fichaje cada vez que comience o finalice su jornada laboral.

[22] API Rest – Conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

En esta sección cada empleado podrá ver qué compañero está trabajando actualmente y tendrá la opción de realizar una llamada desde su terminal IP al teléfono de dicho compañero que se encuentre trabajando sin levantar el auricular, tan sólo haciendo un simple clic en la página web.

Ilustración 35 – Vista de Trabajadores activos

En este caso, estamos viendo la viendo la pantalla "Personal", habiendo iniciado sesión el trabajador con nombre Adrián. Podemos ver las extensiones telefónicas que se encuentran activas, siendo en este caso la extensión 102 (Administrador) y la 103 (María).

Así pues, al pulsar el botón de llamar por ejemplo al administrador, el teléfono de Adrián recibe una llamada. Al descolgar el teléfono escucha un tono de llamada, lo que significa que ya está llamando al teléfono de administrador, tan solo haciendo un simple clic en la interfaz.

Para implementar esta funcionalidad es necesaria la configuración de un fichero de texto con extensión ".call" en la capa Asterisk. Veamos un ejemplo.

En el archivo ".call" está la configuración necesaria para realizar una llamada automática, en concreto el canal de comunicación iría destinado a la prioridad 2, de la extensión 800, dentro del contexto denominado "callme", mediante el canal Zap/1/1XXXXXXXXXXX, con 2 intentos de llamada en caso de no recibir el sistema una traza SIP "Answered" (al descolgar el teléfono), con tiempo de espera de llamada de 30 segundos y un retryTime de 60, esto es el tiempo que se deja entre una llamada y otra.

```
Channel: Zap/1/1XXXXXXXXXXX
MaxRetries: 2
RetryTime: 60
WaitTime: 30
Context: callme
Extension: 800
Priority: 2
```

*Ilustración 36 – Ejemplo de fichero *.call*

6.2.2 Desvío de extensión

La funcionalidad de "Desvio de extensión" conlleva la posibilidad de desviar una extensión telefónica del puesto de trabajo de un empleado desde una sencilla interfaz web y con efectos instantáneos, sin la necesidad de tener que hacer otra operación.

Ilustración 37 – Desvío de llamada

6.2.3 Incidencias (Mini CRM)

Se añade dicha funcionalidad como muestra de una ampliación potencial totalmente personalizada para clientes. Me he decantado por una aplicación simulando un mini-CRM, de ahí su nombre, mediante el cual podemos añadir tickets o partes de incidencias relacionados con un número de teléfono por cliente.

Una de las secciones es la llamada Nuevo cliente, para dar de alta un nuevo canal de incidencias y/o notas relacionadas con un número de teléfono en particular.

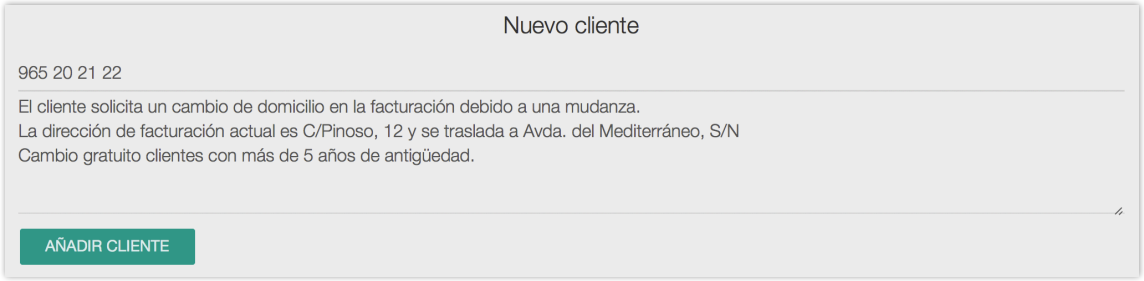

Ilustración 38 – Vista de Mini CRM

Cuando ya tenemos un canal de incidencias abierto y queremos añadir una nota nueva, podemos hacerlo mediante la subsección de búsqueda, mediante la cual buscamos por número de teléfono

y obtenemos todas las notas ordenadas por fecha para poder ver las últimas interacciones del cliente y poder añadir una nota de manera rápida y sencilla.

Ilustración 39 – Vista general de Mini CRM

7. Arquitectura

Una vez que el usuario introduce la información acerca de la configuración que desea establecer mediante la interfaz web, es necesario traducirla en configuración de bajo nivel, en concreto, en lenguaje script de Asterisk.

Para ello se ha decidido establecer una serie de capas mediante las cuales la información de la configuración se va transformando poco a poco en el resultado que se desea obtener. A las capas se le han dado los siguientes nombres:

- **Capa de presentación**. Es la interfaz web donde todo es totalmente amigable para el usuario final, donde no hay perdida ni configuración extraña para un usuario con conocimientos básicos de informática.
- **Capa intermedia**. Se almacenan las configuraciones establecidas por el usuario en diversas tablas de base de datos, manteniendo una armonía enfocada a Asterisk y rellenando los datos proporcionados por el usuario con otros datos extras que serán necesarios en el siguiente paso.
- **Capa Asterisk**. Es la capa de más bajo nivel, donde se encuentra la configuración (en nuestro caso en Real-time) en base de datos escrito en lenguaje de script Asterisk, completamente ilegible para un usuario que no conozca dicho lenguaje.

Es por ello que el paso entre capas facilita la conversión de la configuración del usuario final al lenguaje Asterisk, ya que de esta forma se encuentra todo totalmente modularizado y, en caso de hacerlo directamente, resultaría inviable, ya que aumentaría el acople del código hasta un nivel que habría que borrar toda la configuración en cada paso que diese el usuario final.

7.1 Capa de presentación

Llamamos capa de presentación a la capa con mayor nivel de abstracción, ya que las configuraciones que se añaden son directamente las instrucciones que establece el usuario final en la interfaz de la aplicación. Está compuesta por todas definiciones de extensiones, departamentos, etc. y relaciones entre esos elementos definidas en las pantallas de vistas en el apartado anterior. Esta almacena la configuración en la capa intermedia de manera persistente en base de datos a través del framework Rails.

7.2 Capa Intermedia

A nivel de framework, se ha creado una clase de servicio donde se encuentran todos los métodos que corresponden a este paso, la primera conversión de configuración. A priori puede parecer que se guardan los datos tal cual en las tablas correspondientes, pero por detrás se ha trabajado a conciencia en el modelo de datos para evitar solapamientos y acople entre las diversas funcionalidades de la centralita.

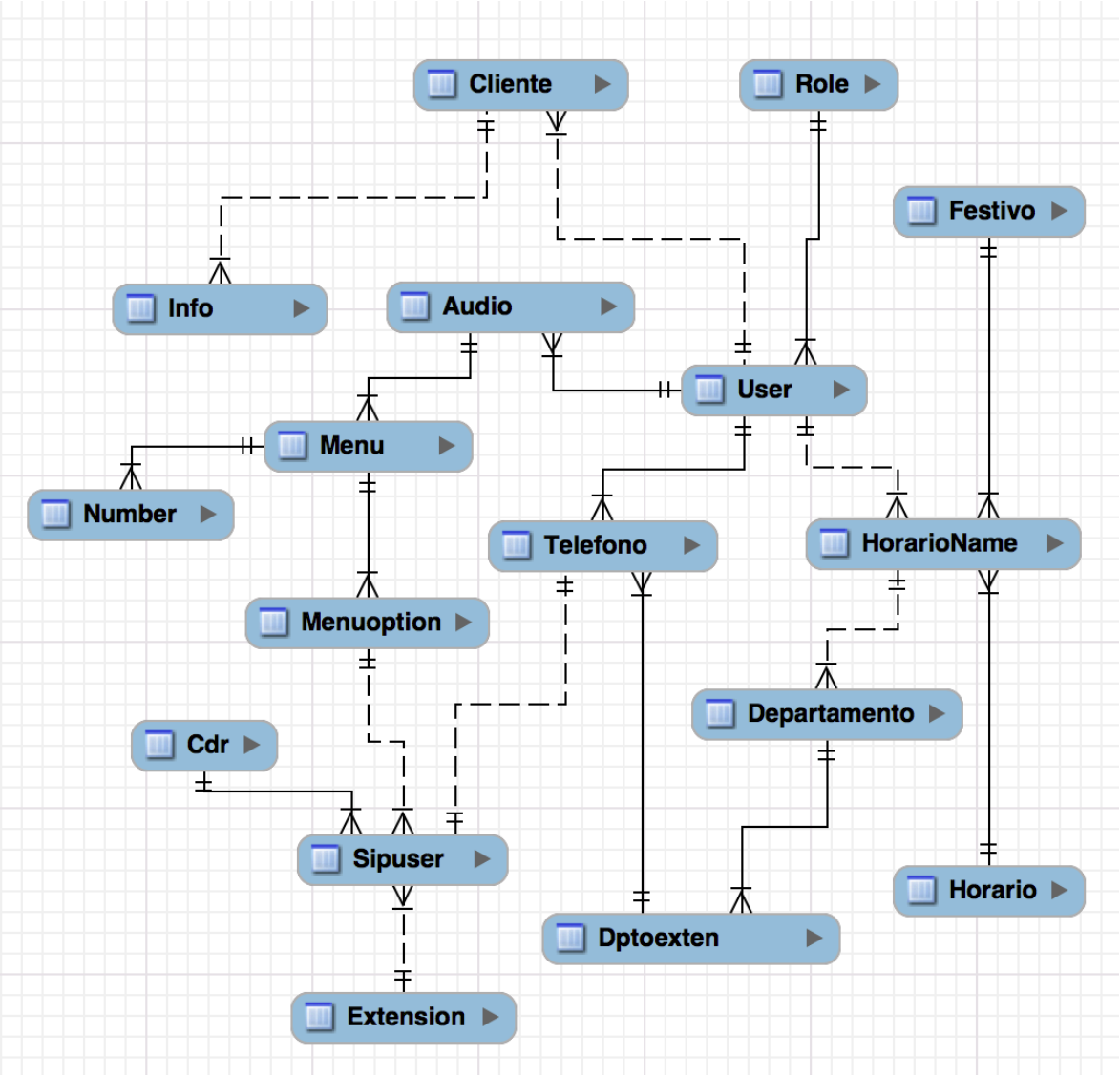

Ilustración 40 – Diagrama del modelo relacional

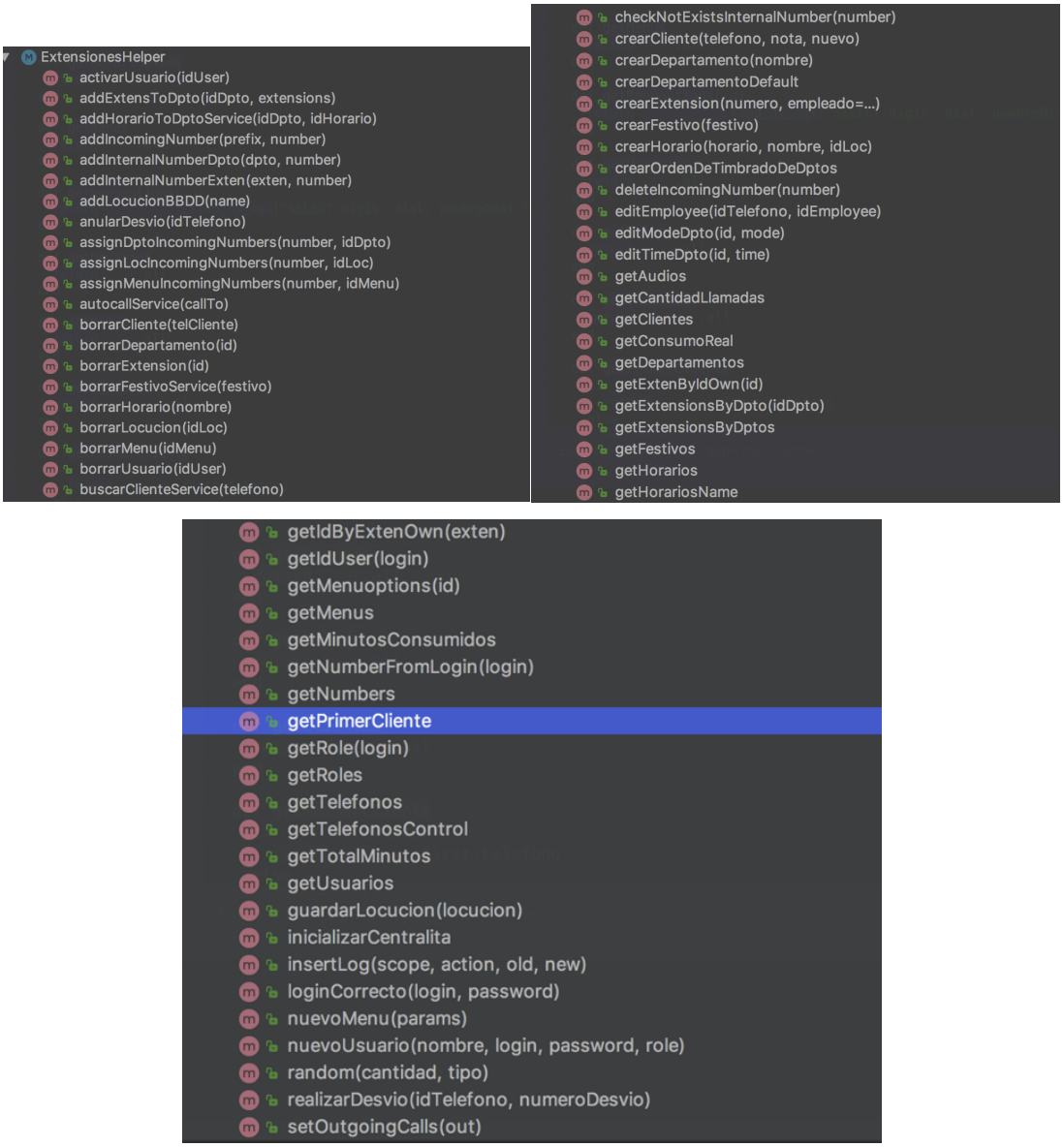

Ilustración 41 – Métodos que atacan al API de Rails

7.3 Capa Asterisk

Es la capa de más bajo nivel, donde reside toda la configuración que Asterisk lee directamente. Como hemos indicado en la sección de tecnologías, originalmente el dialplan está escrito en "/ext/Asterisk/extensions.conf", pero para mayor versatilidad y flexibilidad, se ha configurado en modo *Real-time.*

Por otro lado, Asterisk agrupa las extensiones de la centralita y su configuración por contextos, en este caso se ha prefijado el contexto "PBXTFG[23]", por lo que se resuelve la duda del apartado anterior acerca de "pbxtfg101", que como podemos deducir, es el nombre identificativo de la extensión 101 dentro del contexto PBXTFG.

7.4 Ejempos

7.4.1 Creación de extensiones

Cuando se crea una extensión desde la interfaz web, se realizan los mismos pasos que en los ejemplos que estamos viendo a lo largo del trabajo. De esta manera, los pasos quedan definidos de la siguiente forma:

1. **Capa de presentación**. Se recopilan datos desde la interfaz web.
2. **Capa intermedia**. Se guardan los datos junto a configuración adicional.
3. **Capa Asterisk**. A partir de la capa Intermedia, se crea lenguaje script de Asterisk.

Pero llegados a este punto, ¿qué ocurre cuando creamos una extensión? Esto es algo sumamente importante, pues de ello dependen la mayoría de las funcionalidades. Veamos una imagen del código que se instaura en la tabla extensions.

id	context	exten	priority	app	appdata
34	PBXTFG	pbxtfg101	1	Set	I_USER=${EXTEN}
35	PBXTFG	pbxtfg101	2	Dial	SIP/${EXTEN},120,ort
36	PBXTFG	pbxtfg101	3	Playback	followme/sorry
37	PBXTFG	pbxtfg101	4	Goto	h,1

Ilustración 42 – Ejemplo de la tabla extensions

Como es la primera vez que vemos la tabla en detalle, procedemos a explicar brevemente las columnas y su funcionamiento.

[23] PBXTFG – Nombre del contexto prefijado por defecto en la centralita

- Context. Es el contexto de la centralita para agrupar las órdenes del dialplan (en este proyecto trabajaremos siempre con el mismo contexto).
- Exten. Se refiere al identificador del grupo de flujo, llamado también extensión.
- Priority. Es el número de la prioridad u orden al leer las instrucciones.
- App. Orden de la instrucción.
- Appdata. Se trata de la información para la orden del campo App.

Nota: Cuando hacemos referencia al término extensión, nos referimos a dos conceptos. Por una parte, a un terminal en sí (pbxtfg101), y por otra, al campo de la tabla extensions.conf

Por su parte, cuando en el software de Asterisk lanzamos la instrucción "Dial" junto a "SIP/pbxtfg101, …", lo que ocurre a nivel interno es llamar directamente al teléfono registrado con esa extensión. Por tanto, estos registros de la tabla extensions (imagen anterior), nos permiten llamar a una extensión redirigiendo el flujo de ejecución mediante una instrucción "Goto" a "pbxtfg101,1", donde el número uno del final corresponde al orden de prioridad dentro de la extensión de la tabla extensions, "pbxtfg101".

Posteriormente, podemos ver que, en caso de que la llamada no se conteste, la ejecución pasaría a la instrucción siguiente, que es un "Playback" de una locución de "Lo siento, no se ha podido contactar con el destinatario" (followme/sorry), y después un colgado (Hangup) que es donde terminaría la ejecución de la llamada. Con este trozo de código, podemos llamar a la extensión desde cualquier menú propio, ya sea el de menú selectivo, horario, etc.

Una vez dicho esto, ya podremos entender mejor los siguientes apartados de ejemplo.

7.4.2 Creación de horario

Veamos un ejemplo sobre qué vería el usuario final cuando introduce la configuración para crear un nuevo horario, en donde el rango de días de trabajo sería de lunes a jueves de 9:00 a 20:00 horas y viernes de 9:00 a 15:00 horas.

Ilustración 43 – Menú de configuración

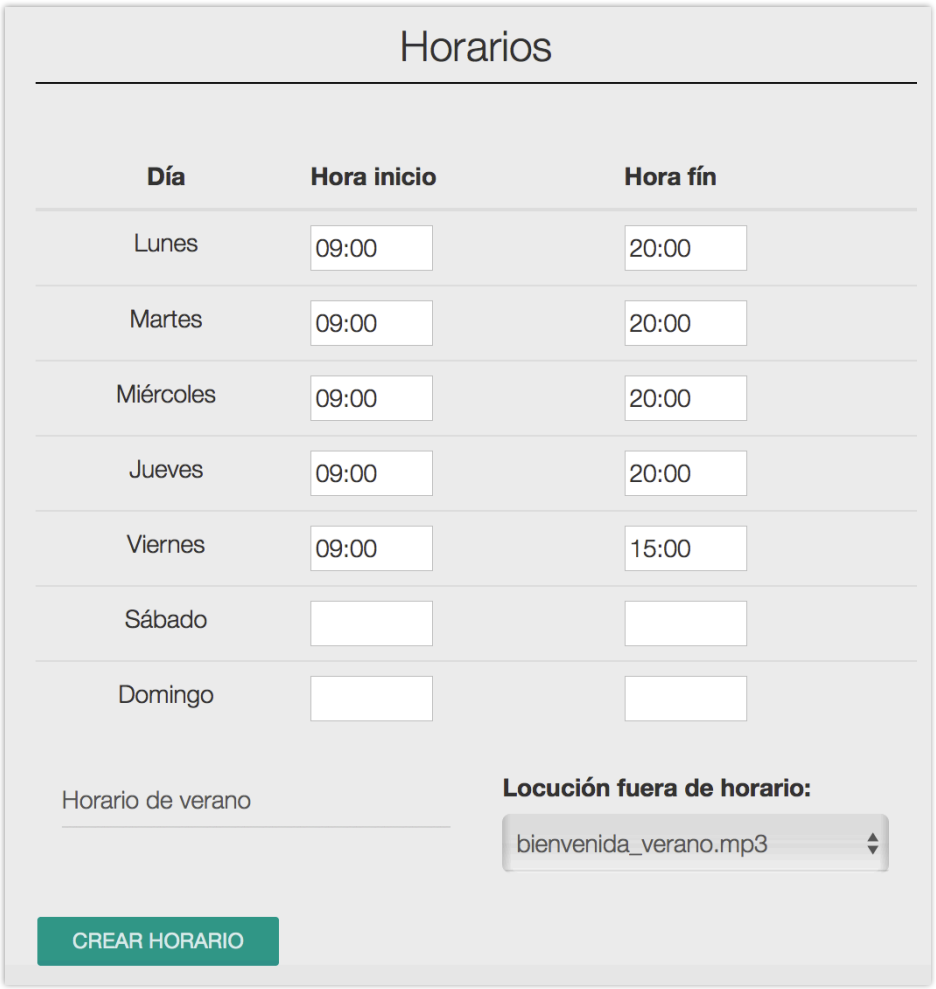

Ilustración 44 – Vista de Horarios

Una vez creado el horario, se puede visualizar de la siguiente forma:

Ilustración 45 – Vista de Horarios seleccionando un horario en concreto

Posteriormente, hay que asociar el horario con algún departamento, como se observa en la siguiente imagen.

Ilustración 46 – Vista de departamentos mostrando opciones de horario y extensiones

Una vez realizada dicha relación, ya tendremos configurado y disponible el horario en la centralita.

Ilustración 47 – Vista de Extensiones telefónicas

Tabla horario

Vemos que se ha creado un registro por día del horario y se ha añadido las tres iniciales de cada día de la semana configurado en inglés. Al crearlo de esta forma se facilitará la conversión a la hora de sincronizar con la capa Asterisk.

id	nombre	dia	day	horainicio	horafin
15	Horario de verano	lunes	mon	09:00	20:00
16	Horario de verano	martes	tue	09:00	20:00
17	Horario de verano	miercoles	wed	09:00	20:00
18	Horario de verano	jueves	thu	09:00	20:00
19	Horario de verano	viernes	fri	09:00	15:00
NULL	NULL	NULL	NULL	NULL	NULL

Ilustración 48 – Tabla Horario

Tabla horarioLocucion

Esta tabla relaciona una locución de bienvenida con un horario con relación uno a uno, lo cual es necesario para facilitar la sincronización y traspasar la información a la siguiente capa.

id	nombre	locucionHorario
7	Horario de verano	1
NULL	NULL	NULL

Ilustración 49 – Tabla HorarioLocucion

Tabla audio

Pertenece a la funcionalidad de añadir locuciones al sistema, pero la añadimos para poder entender mejor la tabla anterior, horarioLocucion. Además, vemos que al insertar las locuciones en base de datos, se le añaden datos extra, como es un alias para poder insertar y localizar las locuciones en el

dialplan una vez realizada esta configuración (horario) o cualquier otra en Real-time, ya que en base de datos el orden de los registros puede variar al visualizarlos según los ordenemos.

id	nombre	alias
1	bienvenida_verano.mp3	loc_1
2	fuera_de_horario.mp3	loc_2
3	lineas_ocupadas.mp3	loc_3
NULL	NULL	NULL

Ilustración 50 – Tabla Audio

Tabla extensions

id	context	exten	priority	app	appdata
255	PBXTFG	hor_7_1	1	Set	DESTINO=s1,1
256	PBXTFG	hor_7_1	2	GotoIftime	09:00-20:00|mon|*|?${DESTINO}
257	PBXTFG	hor_7_1	3	GotoIftime	09:00-20:00|tue|*|?${DESTINO}
258	PBXTFG	hor_7_1	4	GotoIftime	09:00-20:00|wed|*|?${DESTINO}
259	PBXTFG	hor_7_1	5	GotoIftime	09:00-20:00|thu|*|?${DESTINO}
260	PBXTFG	hor_7_1	6	GotoIftime	09:00-15:00|fri|*|?${DESTINO}
261	PBXTFG	hor_7_1	7	Playback	bienvenida_verano
262	PBXTFG	hor_7_1	8	Goto	h,1

Ilustración 51 – Tabla Extension

Cuando el flujo de una llamada entrante realiza un salto a la instrucción “hor_7_1, 1”, se ejecuta el registro 255, en este caso realizaría un “Set” del string “s1,1” en la variable “DESTINO”, que en los registros siguientes se utiliza y se accede como si fuera lenguaje Shellscript: ${DESTINO}.

Desde la capa Intermedia se crea lo que podemos ver en la imagen anterior, esto es, unos registros en la tabla *extensions* que la centralita Asterisk lee directamente.

Por tanto, vemos que se crea una especie de orden “switch-case” dependiendo de qué horario se haya establecido en la configuración, y según nos encontremos en una franja horaria u otra, realizará la acción del “if” o del “else”, parecido a un if-ternario, en este caso no hay “else”.

7.4.3 Creación de menú selectivo

Procedemos a crear un menú que disponga la siguiente configuración:

- Pulsa 1: Llamar a la extensión 101
- Pulsa 2: Llamar a la extensión 102
- Pulsa 3: Llamar a la extensión 103
- Pulsa 4: Llamar a todo el departamento técnico

Por defecto, todos los menús tienen que tener una locución de bienvenida, donde es la propia locución la que da la bienvenida e indica qué dígitos tiene que pulsar el usuario para realizar una acción u otra.

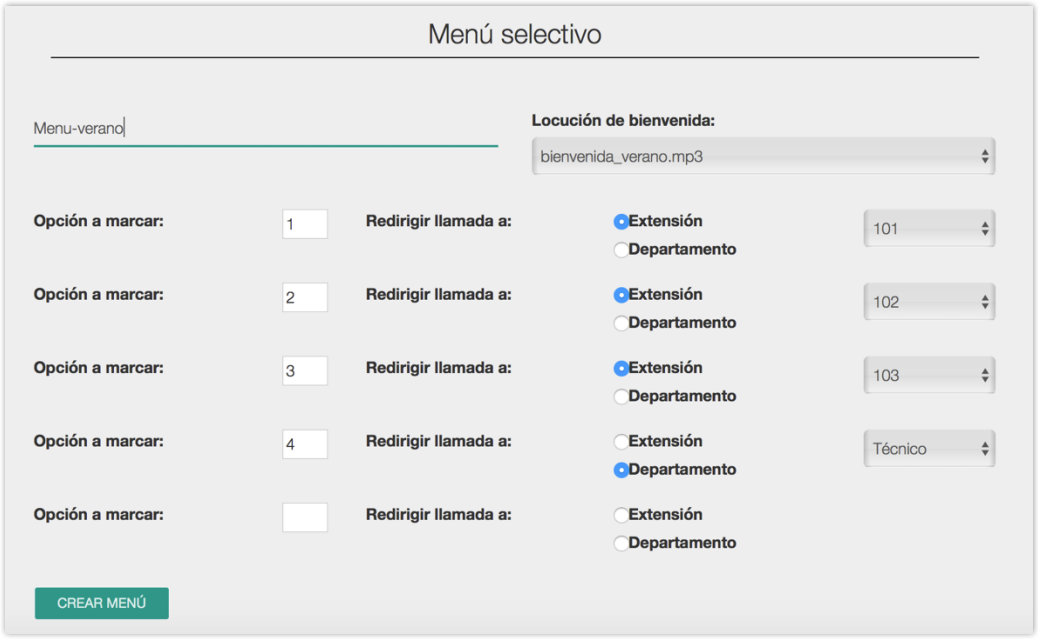

Ilustración 52 – Vista de Menú selectivo

Ilustración 53 – Vista de Menú selectivo seleccionando un menú en concreto

Una vez realizada estas acciones, vamos a ver lo que ocurre en la capa Intermedia.

Tabla menú

Posee el nombre del menú que se inserta junto al id de la locución que se relaciona con éste.

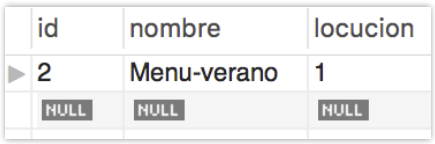

Ilustración 54 – Tabla Menu

Tabla menuOption

Esta tabla contiene las diferentes opciones del menú con su destino. A partir del campo nombreDial (que se establece desde la capa de presentación) podemos obtener la información de la columna "dial", que será lo que utilice la capa Asterisk para insertar la orden de llamada.

id	digit	dial	menu_id	nombreDial
5	1	pbxtfg101	2	101
6	2	pbxtfg102	2	102
7	3	pbxtfg103	2	103
8	4	s2	2	Técnico
NULL	NULL	NULL	NULL	NULL

Ilustración 55 – Tabla MenuOption

Ahora nos surgen dudas sobre, qué significa "pbxtfg101" o "s2", pero lo veremos en el siguiente punto con las tablas de "sipusers" y "extensions".

Tabla extensions

id	context	exten	priority	app	appdata
243	PBXTFG	loc_1_966123123	1	Playback	bienvenida_verano.mp3
244	PBXTFG	loc_1_966123123	2	Goto	s2,1
250	PBXTFG	Menu-verano	1	Read	DIGIT_DIALED,"bienvenida_verano",1,,1,5
251	PBXTFG	Menu-verano	2	Gotoif	$[$["${DIGIT_DIALED}" = "1"]]?pbxtfg101,1:
252	PBXTFG	Menu-verano	3	Gotoif	$[$["${DIGIT_DIALED}" = "2"]]?pbxtfg102,1:
253	PBXTFG	Menu-verano	4	Gotoif	$[$["${DIGIT_DIALED}" = "3"]]?pbxtfg103,1:
254	PBXTFG	Menu-verano	5	Gotoif	$[$["${DIGIT_DIALED}" = "4"]]?s2,1:

Ilustración 56 – Tabla Extensions

Nos encontramos otra vez con la tabla "extensions" y esto se debe a que toda la configuración de nuestra capa Asterisk reside en dicha tabla.

Vemos que se crea una extensión específica para agrupar la configuración de menú, con sus correspondientes prioridades, que darán el orden de ejecución y posteriormente, las instrucciones con su información, también configurada desde la interfaz web en primera instancia por el usuario.

En el registro con id 251 vemos que, si la variable DIGIT_DIALED es igual a 1, realizará un "Goto" a la extensión "pbxtfg101,1".

7.5 Sincronización entre capas

Una vez explicadas cada una de las capas por separado, cabe destacar que no siempre es conveniente realizar la sincronización, por lo que para sincronizar la capa Intermedia y capa Asterisk se creó un método único que, después de varias pruebas, se optó por realizar la llamada de sincronización según lo requiriese la funcionalidad que estuviese en ejecución en dicho momento. De esta forma, conseguimos una modularidad total para independizar las funcionalidades.

Al igual que en la capa intermedia, poseemos una clase de servicio de segundo nivel, donde tenemos todos los métodos que afectan directamente a la base de datos, en concreto a la tabla *extensions* y que escriben directamente el código de script Asterisk.

Estos métodos únicamente son accedidos desde la clase de servicio de la capa Intermedia, y esta a su vez desde la capa de presentación.

Veamos la definición de dicha clase con los diferentes métodos de los que se ha precisado para establecer toda la configuración en la etapa final del proyecto.

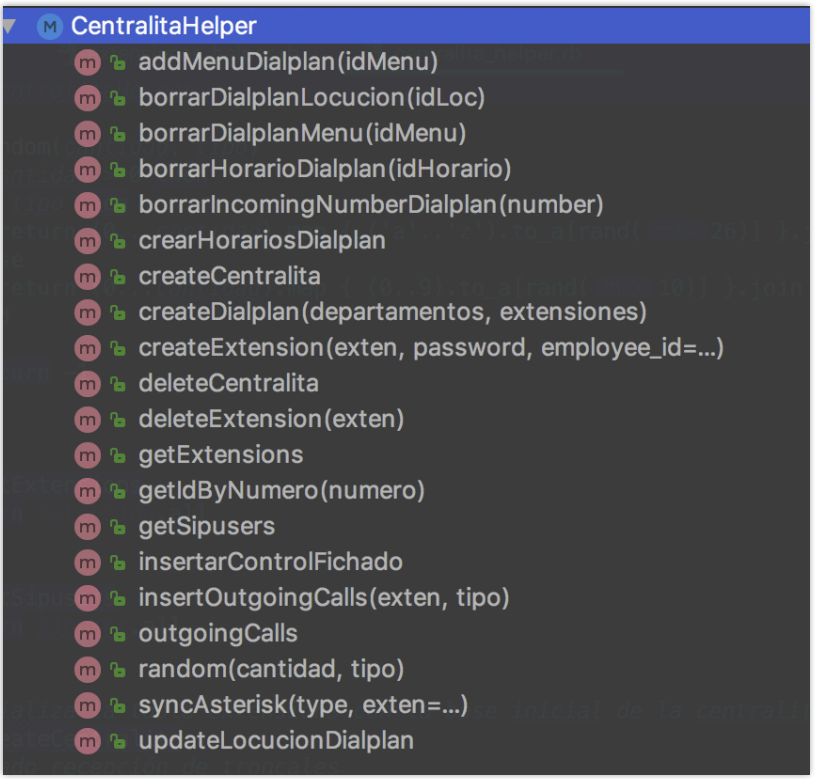

Ilustración 57 – Métodos que atacan al API de Rails de la capa Asterisk

A continuación, vamos a proponer un par de ejemplos que iremos siguiendo durante la explicación de todas las capas de la aplicación, de forma que podremos seguir más fácilmente la configuración cuando nos encontremos en las capas de más bajo nivel. Por ejemplo, crearemos un horario

telefónico de centralita y un menú selectivo con locución de bienvenida. Estas funcionalidades son dos de las que poseen más complejidad a nivel interno, ya que combinan casi todos los elementos de la centralita: extensiones, departamentos, locuciones, etc., que a su vez poseen otra lógica interna.

Damos por hecho que en la configuración se han creado previamente 3 extensiones telefónicas asociadas ya a sus empleados (101: Adrián, 102: Administrador y 103: María) y unos departamentos de ejemplo (General, Técnico y Administración)

Nota: El departamento de administración no corresponde al administrador, sino al grupo de empleados que llevan labores administrativas de la empresa.

8. Seguridad

Para la comunicación mediante el protocolo SIP, Asterisk utiliza por defecto el puerto 5060, lo cual es abierto nada más finalizar su instalación. Si pusiéramos un receptor de peticiones de tipo INVITE (tipo de petición del protocolo SIP) veríamos que recibimos cantidad de peticiones de atacantes de la red.

8.1 ¿Por qué atacan Asterisk?

Asterisk es utilizado por muchas empresas como centralita virtual para su telefonía y, por tanto está interconectado con algún proveedor IP mediante un Trunk SIP con salida vía telefónica al exterior. Con esto nos referimos a que las empresas pueden realizar llamadas a números convencionales a través de un proveedor.

Muchas empresas hoy en día aún utilizan contraseñas débiles para proteger su interfaz web y los atacantes son conscientes de ello, por lo que dedican tiempo a realizar ataques y programar bots[24] de ataque masivo. Esto se realiza enviando peticiones INVITE para intentar registrar extensiones al azar en centralitas de la red, con la esperanza de aprovechar alguna mala configuración de Asterisk y poder sacar llamadas de manera rápida y gratuita.

Hay un gran mercado detrás de todos estos fraudes telefónicos, aunque no vamos a entrar en detalle. Por esto, se han limitado las llamadas salientes prohibiendo las llamadas a números de tarificación adicional (803, 806, …) y números Premium, cuyos propietarios suelen estar detrás de los ataques para llamarse a sí mismos y cobrar por las llamadas recibidas a un precio elevado.

[24] Bots – Programas informáticos que realizan ciertas instrucciones repetitivas de manera automática

8.2 Servidor

Como primera medida de seguridad destaca es asegurar todo lo posible el servidor, porque como hemos dicho anteriormente, una vez finalizada la instalación de Asterisk empezaremos a recibir peticiones de atacantes cuando detecten nuestro puerto 5060 abierto al exterior.

Para ello se han establecido las siguientes medidas de seguridad:

- Firewall IP Tables
- Fail2Ban (Baneo de las IPs que realizan intentos de conexión fallidos al servidor)
- SipCheck (Programa escrito en Python que lee el log de Asterisk detectando los intentos de conexión y registro fallidos, baneando la IP de origen en caso reincidente)
- Cambio de puertos predeterminados en SSH y servidor web
- Deshabilitar acceso Root[25] por SSH[26]
- Habilitar HTTPS[27] con certificado autofirmado

Como añadido, destacar que existe un proxy SIP llamado Kamailio que sirve para filtrar y redirigir llamadas en caso de gran cantidad de tráfico, lo cual es necesario para un operador de telefonía, no para una empresa.

8.3 Asterisk

La configuración principal en Asterisk se halla repartida en varios ficheros (asterisk.conf, sip.conf, res_mysql_config.conf, extconfig.conf, etc.), pero podemos destacar como el más importante en cuanto a la seguridad el fichero sip.conf, ya que posee unos determinados parámetros, que de estar mal configurados, pueden permitir la entrada a la centralita a intrusos, incluso registrar teléfonos y realizar llamadas sin tener ni un solo permiso.

Los parámetros básicos para securizar Asterisk son:

- Guest_allow = no

  Permite a extensiones, no dadas de alta en el sistema, registrarse y realizar llamadas libremente.

[25] Root – Rol de superadministrador
[26] SSH – Secure Shell es un protocolo que facilita las conexiones remotas entre dos sistemas utilizando la arquitectura cliente/servidor
[27] HTTPS – Protocolo HTTP sobre un túnel cifrado TLS o SSL

- Canreinvite = no

Permite recibir peticiones tipo invite, mediante las cuales el atacante puede obtener información y, después de un tiempo, conseguir acceder a la centralita.

9. Testing

9.1 Pruebas unitarias con Ruby on Rails

Rails nos permite realizar test unitarios, por lo que se han implementado de uno a tres test por cada uno de los métodos de las clases de la capa de servicio.

Al realizar la CI, se ha configurado el fichero travis.yml para que una vez se confirme el *Pull request* de la rama de desarrollo con la rama develop, se lancen todos los test unitarios.

De esta forma, a medida que se han ido ampliando funcionalidades, no se ha dejado de probar la aplicación en ningún momento, lo cual nos da una fiabilidad alta de consistencia.

También he de mencionar que han habido test a lo largo del desarrollo que han fallado. Esto me ha permitido realizar los ajustes necesarios para solventar los fallos y hacer el software cada vez más robusto.

```
# setOutgoingCalls
test  name "establecer llamadas salientes" do
  primer = ['f', 'm']
  segun = ['f', 'r', 'i']
  out = {'101' => primer, '102' => segun}
  crearExtension('101')
  crearExtension('102')
  setOutgoingCalls(out)
  result = Telefono.find_by_numero('101').movil

  assert_equal true, result
end
```

Ilustración 58 – Test unitario 1

```
test  name "create extension 202" do
  createExtension( exten '202',  password 'pass')
  ext = Extension.where("exten like 'pbxtfg202' AND app = 'Dial'")
  result = ext[0].appdata
  response = 'SIP/${EXTEN},120,ort'

  assert_equal response, result
end
```

Ilustración 59 – Test unitario 2

```ruby
# ISS19 - Horario con departamentos
test "addHorarioToDpto" do
  dptoName = "oficina"
  horarioName = "verano"
  horario = [nil, ["09:00", "18:00"], ["", ""], ["", ""], ["", ""], ["", ""], ["", ""], ["", ""]]
  crearDepartamento(dptoName)
  crearHorario(horario, horarioName, idLoc nil)
  idHor = Horario.find_by_nombre(horarioName).id
  idDpto = Departamento.find_by_nombre(dptoName).id
  addHorarioToDptoService(idDpto, idHor)
  result = Departamento.find_by_nombre(dptoName)
  expected = Horario::HOR + idHor.to_s
  assert expected, result.horario
end
```

Ilustración 60 – Test unitario 3

9.2 Pruebas de funcionales: Raspberry PI[28]

Las pruebas funcionales las he realizado a medida que iba avanzando el proyecto, directamente sobre el dispositivo Raspberry Pi, con el fin de asegurar que todo iba correctamente y que se obtenía la configuración aplicada mediante la interfaz web en la centralita de Asterisk.

[28] Raspberry Pi – Ordenador de placa reducida

10. Instalación

10.1 Introducción

Para desarrollar este proyecto he necesitado realizar una instalación previa de cierto software e infraestructura. He utilizado una Raspberry Pi model B para instalar Asterisk y Ruby on Rails para correr la aplicación.

Por ejemplo, en un entorno real se podría utilizar la Raspberry Pi con la instalación de este proyecto y poder dar servicio telefónico con todas las funcionalidades indicadas a un coste realmente bajo.

10.2 Instalación del servidor

Como hemos mencionado en el punto anterior, he optado por la utilización de una Raspberry Pi y con ella el sistema operativo Raspbian

Hay poco que destacar en cuanto a la instalación del sistema operativo, pues este modelo de Raspberry posee una tarjeta SD a modo de disco, la cual hay que formatear y realizar la instalación del sistema operativo desde una computadora externa a modo de imagen.

Una vez realizada la instalación, solo nos queda insertar la tarjeta SD en la Raspberry e iniciarla.

10.3 Instalación de Asterisk PBX

Asterisk es un programa open source programado en lenguaje C, por tanto, al descargarlo de la página oficial obtenemos un fichero tar.gz. El siguiente paso es descomprimirlo y entrar en la carpeta ejecutamos los siguientes comandos:

```
1) ./configure
2) make menuselect
3) make
4) make install
5) make samples
6) make config
```

Realizando estas acciones configuramos la instalación de Asterisk, y mediante el comando “make menuselect”, seleccionamos los módulos que deseemos.

Una vez instalado, podemos arrancarlo a modo de servicio “/etc/init.d/Asterisk start” y entrar en el CLI de Asterisk mediante la instrucción: “Asterisk –r”.

Como caso opcional, para ver más información y detalle en la consola de lo que ocurre en la centralita, podemos aumentar el nivel de verbose hasta 5, añadiendo la letra “v” después del parámetro “-r”, quedando de la siguiente forma: “asterisk –rvvvvv”.

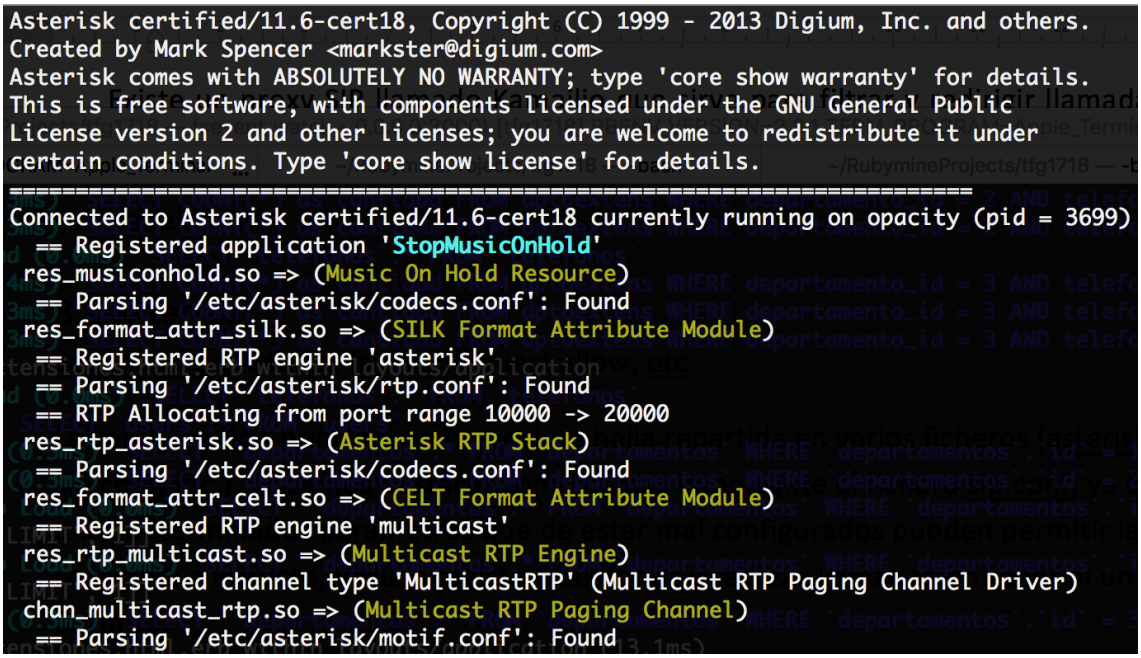

Ilustración 61 – CLI de Asterisk

10.4 Instalación de Ruby + Rails

Para la instalación de Ruby y su framework Rails introducimos los siguiente comandos:

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Una vez instalado Homebrew, procedemos a instalar RVM:

```
\curl -L https://get.rvm.io | bash -s stable –ruby
```

Una vez realizado esto, podemos comprobar la versión que tenemos instalada de ruby:

```
ruby -v
```

El framework Rails lo instalamos mediante una gema de ruby:

```
gem install rails
```

Finalmente podemos verificar la versión de rails y ya tendremos el framework listo para usarlo:

```
Rails –v
```

11. Conclusiones y mejoras

Haciendo una retrospección, vemos que se han abarcado un número superior de funcionalidades al estimado a principios del proyecto, esto se debe a que el mundo de la VoIP es inmenso y con creatividad, podemos obtener infinidad de soluciones de cualquier ámbito, por ejemplo, un CRM que detecta las llamadas de la centralita, un control de fichajes, llamadas automáticas mediante la interfaz web, etc. Obviamente se hizo una valoración y estimación de tiempo para realizar las funcionalidades que entrarían en el plazo del desarrollo, dejando fuera muchas otras.

A continuación, vamos a nombrar algunas de las funcionalidades que se han dejado en el backlog para posibles mejoras en un futuro.

11.1 Grabación de llamadas

Funcionalidad que permite grabar las llamadas de cada una de las extensiones de la centralita para su escucha posterior.

11.2 Voicemail

Envío por email de un buzón de voz, grabado por un cliente cuando no se contesta su llamada.

11.3 Fax2mail

Funcionalidad que permite recibir un fax y enviar el adjunto como un pdf vía email. Esto se realiza con el códec SIP T.38.

11.4 Mail2fax

Funcionalidad inversa a Fax2Mail que consiste en enviar por email un adjunto para imprimirlo en un fax. También se realiza con el códec SIP T.38.

11.5 Llamadas vía WebRTC

Funcionalidad que permite realizar videollamadas mediante navegador con la webcam y está ligado a la centralita Asterisk.

11.6 Facturación

En el caso de que se instalase el software para un grupo de empresas, y dentro de la centralita poseer el control simultáneo de todas ellas, se podría realizar una funcionalidad que calculase la facturación de cada empresa dentro del grupo. Esta opción no se ha planteado seriamente ya que se salía del enfoque general del proyecto.

11.7 Domótica

En este apartado podemos perdernos literalmente, ya que podemos controlar lo que quisiéramos mediante la centralita virtual. Entre muchos ejemplos podemos destacar el control de persianas y aparatos eléctricos, realizando una llamada a una extensión que configuremos en la centralita, lanzando un script y mediante la Raspberry Pi por el puerto GPIO, controlar aparatos de domótica interconectados a una vivienda. Esto correspondería un poco a la sección IoT, por lo que se ha optado por dejar fuera del alcance del proyecto.

11.8 Planes de marcado inteligente para call-centers

Las compañías tipo call-center utilizan a diario planes de marcado automático para realizar llamadas de telemarketing. Esta funcionalidad también se puede elaborar a través de Asterisk, aunque estaría muy enfocada a este tipo de empresas.

12. Referencias y bibliografía

12.1 Proyecto

- Aplicación centralita virtual: https://github.com/agosalvez/tfg1718

12.2 Tecnologías y herramientas utilizadas

- Trello: https://trello.com
- GitHub: https://github.com
- GitHub Pages: https://pages.github.com
- CI: https://travis-ci.org
- Ruby: https://www.ruby-lang.org/es/
- Raspberry PI: https://www.raspberrypi.org/
- Raspbian: https://www.raspberrypi.org/downloads/raspbian/
- Asterisk: https://www.asterisk.org/
- PHP5: http://php.net/
- Javascript + jQuery: https://jquery.com/
- MySQL: https://www.mysql.com
- MySQLWorkbench: https://www.mysql.com/products/workbench/
- RubyMine by JetBrains: https://www.jetbrains.com/ruby
- Visual Studio Code: https://code.visualstudio.com/
- SSH: https://www.ssh.com/ssh/
- Shellscript: https://www.shellscript.sh/

12.3 Material de apoyo

- Ruby Guides: https://guides.rubyonrails.org/
- Active Record (ORM): https://guides.rubyonrails.org/active_record_basics.html
- API RESTful in Rails: https://scotch.io/tutorials/build-a-restful-json-api-with-rails-5-part-one
- Kanban – Artículo de Atlassian: https://es.atlassian.com/agile/kanban
- Sinologic: https://www.sinologic.net/
- FreePBX: https://www.freepbx.org/
- Elastix: https://www.elastix.org/
- AsteriskNow: https://www.asterisk.org/downloads/asterisknow
- Asterisk ARI: https://github.com/svoboda-jan/asterisk-ari
- Grandstream: http://www.grandstream.com/
- Apuntes de Metodologías Ágiles de Desarrollo Software de la UA

- Apuntes de GIT de la UA: https://github.com/domingogallardo/apuntes-mads/blob/master/practicas/01-introduccion-play/comandos-git.md
- Apuntes de Seguridad en el Diseño Software de la UA
- Apuntes de Planificación y Pruebas de Sistemas Software de la UA
- Apuntes de Aplicaciones Distribuidas en Internet de la UA
- Apuntes de Diseño de Bases de Datos de la UA